Specification & Design of Real-time Systems

FA_07_SE_545: ERAU TEAM BLUE

Project Description
Automatic Production Environment

Christopher Griffis
Steve Harvey
Leonardo Matos
Jason McGuire
Sean Pfeifer
Caylyne Shelton

Document Information

Category	Information
Document ID	Project Description: Automatic Production Environment
Document Owner	FA_07_SE_545: ERAU TEAM BLUE

Revision History

Date	Version	Description	Author
9/20/07	0.1	Initial document template	Christopher Griffis
12/10/07	0.2	Document high level structure	Christopher Griffis
12/11/07	1.0	Document Authoring	Christopher Griffis

Document Approvals

Role	Name	Signature	Date
Project Sponsor	A. Kornecki		12/11/07
Project Review Group	FA_07_SE_545		12/11/07
Project Lead	Christopher Griffis		12/11/07
Requirements Lead	Sean Pfeifer		12/11/07
Hardware Lead	Jason McGuire		12/11/07
Design Lead	Caylyne Shelton		12/11/07
Programming Lead	Steve Harvey		12/11/07
Quality Manager	Leonardo Matos		12/11/07

Table of Contents

Table of Figures

List of Tables

Project Description
Automatic Production Environment

1. Introduction

SE545: Specification and Design of Real-time Systems is a graduate level software engineering course at Embry-Riddle Aeronautical University, Daytona Beach Campus. As of the fall of 2007, this course requires the production of "software artifacts representing the core operational part of a selected system." The project Automatic Production Environment aims to fulfill these course requirements through a well defined real-time approach to software development.

This document discusses the project as an overall effort, first introducing the description, purpose, and project scope. The team organization is discussed, leading into a comprehensive discussion of the methodologies used in developing both the system and the real-time software. This includes a discussion of the software development process used, the steps followed in implementing that process, and the platform and operating system. Also, included are some design decisions, notations used, and the lessons learned for the project.

1.1 Description

The project centers around a system labeled the Automatic Production Environment, affectionately abbreviated as "APE." The APE (Figure 1) is an electromechanically driven conveyor belt system built out of small generic-use modeling blocks, (similar to Lego building blocks). The device is meant to simulate an industry control system, imitating the operation of an airport luggage conveyor belt system with a security scanning machine. The APE will follow specified behavioral requirements to transfer a payload/luggage item along an 'L' shaped conveyer belt and "scan" the package (simulating a security scan).

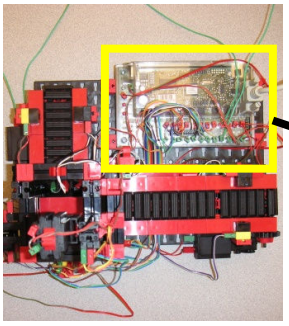

Figure 1: APE Hardware

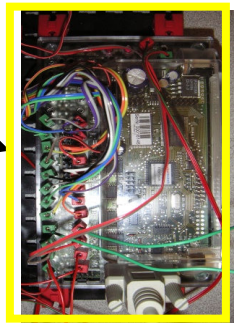

Figure 2: ROBO Interface

Figure 3: Arcom Hardware

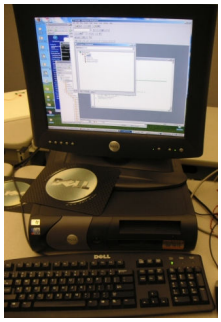

Figure 4: Tornado IDE

The overall arrangement has the footprint of the average college textbook (Figure 1). Small electric motors and sensors carefully arranged in the system are wired to an onboard embedded interface system (Figure 2: ROBO Interface). A real-time computing system (Figure 3: Arcom Hardware) communicates over a serial cable with the embedded interface system to control the operation of the APE. The bootable software image that is downloaded to the Arcom box is developed, tested, and debugged in the Tornado IDE (Figure 4). The system has the ability to meet specified transition, motion, and timing requirements provided in the user needs document. Unlike typical academic software projects, this project looks at the problem from a *system* perspective, and will serve to emphasize real-time software principles.

1.2 Purpose

The purpose of this project is perform a well organized, group-based effort in creating a successfully working Automatic Production Environment, as well as implement and experience a well-documented progression through

the various steps of a real-time oriented software development lifecycle. It has been said that this is a "project" focused effort, not a "product" focused effort; the underlying intent is not only to have a functioning APE, but also to use creation of the APE as a means of practicing sound software engineering principles in a real-time context.

1.3 Project Scope

The project was performed by a team of six software engineering graduate students at Embry-Riddle Aeronautical University. The total effort roughly occurred over a three month period, starting in early September of 2007 running until completion in mid-December of 2007. All primary development was done on a single computer with the integrated development environment, connected to a single Arcom real-time box running the VxWorks operating system. There was a single prototype system constructed from the Fischertechnik modeling blocks. The total effort for the projet was 449.75 hours, averaging 74.96 hours of effort/person (σ = 25.57 hours).

The project scope is summarized in the following Is/Is Not list:

Table 1: Project Scope: IS / IS NOT List

This Project IS	This Project IS NOT
This project **is** a small scale project consisting of 6 team members.	While it may address this item, this **is not** a project on software process.
This project **is** meant to fulfill the user needs within one academic (ERAU) semester.	While it may address this item, this **is not** a project on software documentation.
This project **is** an experience meant to serve as an academic tool.	While it may address this item, this **is not** a project on software project management.
This project **is** intended to emphasize the principles of real-time system development and operations.	While it may address this item, this **is not** a project on software requirements.

2. Team Organization

2.1 Stakeholders

- Sponsor: Dr. Andrew J. Kornecki, Embry-Riddle Aeronautical University
- Members of TEAM BLUE
    - Christopher Griffis (Project Lead)
    - Steve Harvey (Programming Lead)
    - Leo Matos (Quality Lead)
    - Jason McGuire (Hardware Lead)
    - Sean Pfeifer (Requirements Lead)
    - Caylyne Shelton (Design Lead)

2.2 Team Roles

2.2.1 Team Lead, Planning/Requirements Lead (Christopher Griffis)

The Team Leader is responsible for maintaining the process and team discipline. The Team Leader leads the team and ensures that engineers report their process data and complete their work as planned. He tracks and reports the team progress. He facilitates team meetings and keeps record of meeting times and Action Items.

The Requirements Manager primary role is to lead in the requirements phase. He should have an excellent understanding of requirements analysis and will be leading the effort to create a SRS.

2.2.2 VxWorks Coding Lead (Sean Pfeifer)

The VxWorks Coding Lead is responsible for all coding of the VxWorks real-time control aspects. He will use the SDS as a basis for his code and will lead the effort for producing the code and code review.

2.2.3 Design Lead (Caylyne Shelton)

The Design Manager primary role is to develop a high/low level design for the development of the software product. She will lead the effort to produce a SDS.

2.2.4 Webserver Coding Lead (Steve Harvey)

The Webserver Coding Lead is responsible for all coding of the webserver aspects. He will use the SDS as a basis for his code and will lead the effort for producing the code and code review.

2.2.5 Quality Lead (Leonardo Matos)

The Quality Manager maintains the team timelogs and TSP data, and is heavily involved in all the inspections through the development process. His responsibility is to track development problems or issues and to report to the Team Leader. The Quality Manager will prepare the test plan and coordinate the testing phases.

2.2.6 Hardware Lead (Jason McGuire)

The Hardware Manager is responsible for ensuring that the hardware equipment operational and up-to-date. He will perform hardware maintenance tasks when needed and will also help with identify hardware requirements throughout development. He will also assist in the coding of the webserver and VxWorks integration.

3. Methodology

Team Blue used a self-developed software process based on the waterfall software development lifecycle. The project began with the development of the project plan, understanding the high level user requirements and taking into consideration available human resources and time constraints. The resulting documentation artifact (the Software Development Plan or SDP) is described in 3.2.1. The next step was to focus all efforts on a concisely, completely, correctly, and consistently defined model of the problem, documented in the form of a requirements specification (described in section 3.2.2). The requirements specification was considered a critical step, with an inspection process and concomitant document revision to ensure sufficient quality. Once the Software Requirements Specification was baselined, the design process began. Taking into consideration all the issues identified in the requirements, both a comprehensive design specification and test plan were created. The design specification (described more in section 3.2.3) was also considered a critical step necessitating an inspection process. After the design was completed and baselined, the coding process began. A code base was developed, with an intermediate demonstration of product functionality. Finally, upon completion of the coding and successful testing, a project postmortem (including a project summary, TSP analysis, and APE user manual) was performed.

3.1 Development Environment

The development Environment consists of 5 major parts: the APE model (Figure 1) with motors and sensors connected to the ROBO interface (Figure 2), the Arcom Box (Figure 3) containing the processing execution platform, the VxWorks operating system running the real-time image on the Arcom box, the GoAhead webserver bundled into the bootable image, and the Tornado IDE (Figure 4) providing the software development facilities.

3.2 Development Process

As stated in earlier, the waterfall software development lifecycle was used. However, it should be mentioned that various other aspects were used to improve quality and facilitate team management logistics. A modified Team Software Process was implemented when administrating team activity. This included keeping standardized time logs and meeting records. Regular team meetings promoted team cohesion and collaboration, and comprehensive meeting records allowed for insights occurring during meetings to be documented and used later. The “Blackboard”

collaboration tool was also used as a digital meeting place, where team members openly and frequently exchanged ideas and dialogue, greatly promoting project progress.

A documented set of procedures and configuration management policies allowed for consistent and effective team interaction when creating project artifacts. For example, a consistent document format was implemented in every project artifact; each artifact has the same cover page layout, and each provides the same kind of document information, signature page, and preface identifying the applicability of any industry recognized standard being followed. Also, each major artifact has the same type of table of contents, and list of figures and tables. Also, the header on each page is populated with the course name, project ID, team name, document version, and date of last baseline. Other configuration management procedures in place can be found in the Software Development Plan documentation.

3.2.1 Planning

The entire effort of planning is documented in the Software Development Plan (SDP), which loosely follows the IEEE Std. 1058-1998, "IEEE Standard for Software Project Management Plans." It begins with a project summary, including the project purpose, document purpose, stakeholders, scope and objectives. Next is a discussion of the system high level requirements and project deliverables, followed by a section on the project organization, discussing team role responsibilities, and available resources. Process plans such as estimation plan, work breakdown structure and project schedule are also included. Moreover, a brief risk management plan and discussion of the process plan (such as lifecycle selection), as well as a description of the process model are addressed. Finally documented in the software development plan (SDP) was a discussion of the configuration management plan and quality assurance plan. The final version of the SDP contains a brief TSP analysis and the complete listing of all meeting records.

3.2.2 Requirements

The requirements are concisely documented in the Software Requirements Specification (SRS), which loosely follows the layout suggested by IEEE Std. 830-1998, "IEEE Recommended Practice for Software Requirements Specifications." The SRS introduces the problem by first describing the purpose of the project and document, as well as the scope and an overview of the SRS contents. It describes the requirements from the product perspective using a formalized narrative, supported by models and interface description. The requirements contain an explanation of all the required product functions, user characteristics, and assumptions. The heart of the SRS, the specific requirements, contains all the system, hardware, software, and interface requirements.

Included in the Software Requirements Specification are sequence diagrams, an entity-relationship diagram, a data dictionary, and a requirements traceability matrix. State charts identify the state behavior and modes of operation, and process specifications delineate the required functional capabilities of the APE (including timing information and parameter details for each process). Data flow diagrams detail the execution behavior of the processes, and illustrate various real-time aspects of inter-process communication. Use case scenarios help explain the operation from a user interaction perspective, and identify the possible errors that must be accounted for during APE operation.

3.2.3 Architecture and Design

The architecture and design phase resulted in the Software Design Specification/Description (SRS or SDD), which loosely follows the layout suggested by IEEE Std 1016-1998, "IEEE Recommended Practice for Software Design Descriptions." First discussed are the introductory aspects, reiterating the project purpose, system overview scope, and document overview. Next explained are the system architectural components and mechanistic design aspects (e.g. system states and entity relationships). The main part of the document is a comprehensive declaration of the detailed design aspects, including the process design specifications, system modes, and the web interface design.

For each high level process to be implemented in the system, the process design specifications declare the name and description for each process, supported by the list of inputs and outputs of each process. Subprocesses are

referenced, and the main algorithm is documented. The process design specifications also include pseudocode for implementation in the specific language (in this case, VxWorks C) and declare timing behavior for meeting complex real-time constraints. The process design specs conclude with identification of possible error states or conditions.

The system operates in two modes: automatic mode and user control mode. In automatic mode the system shall behave according to the behaviors declared in the requirements; user control mode allows the user direct control over each motor. The design document explains the exact mechanics of the two modes the system will operate in, including details of the alternate data flows for each respective mode.

The system is controllable through a web interface, which must interact with the downloaded real-time boot image on the Arcom hardware to fulfill web control requests. The details of this interface and the design behind its interaction with the bootable software image on the Arcom box are explained in the Web Interface section of the design document, which provides design mockups with descriptions and pseudocode for the webserver.

The SDS identifies many important design decisions in place for the Automatic Production Environment project. The webserver, sensing functions, and motor actuating functions are encapsulated into separate modules, allowing for higher cohesion and lower coupling than alternative configurations. Multiple threads of operation were used to implement concurrent aspects of the sensor polling and motor actuation, as well as the webserver interaction with the onboard system. Originally, the design provided for inter-process communication in the form of message queues. Later, it was determined that a semaphore-protected shared data model (in the form of global variables) would help mitigate communication synchronization issues. Watchdogs are in place to ensure timeouts are enforced, in response to possible error states identified during requirements phase.

3.2.4 Coding

The coding phase involved implementing the design documented in the SDS. The VxWorks C files and the Webserver ASP files are developed on the host environment, compiled, and downloaded onto the target (Arcom Box). When the coding process is successfully completed and all bugs are removed, the code is analyzed for size and complexity. This information, along with the code itself, is documented in the VxWorks and Webserver Code Document.

3.2.5 Testing

The test phase has two major aspects, the test plan creation, and the actual project testing. After the final version of the SRS was baselined, the Automatic Production Environment Test Plan was drafted and baselined. The test plan document loosely follows the layout suggested by IEEE Std. 829-1998, “IEEE Recommended Practice for Software Test Documentation,” and has the pro forma introductory information, identifying the document purpose, project stakeholders, scope and overview. It outlines the various test items, software risk issues, and features to be tested. A brief description of the test approach is given, as well as the pass/fail criteria and the entry and exit criteria. The document concludes with a list of the actual test cases, which appear as testing scripts and parallel the use cases identified in the requirements document.

The second part of testing is the execution of the prescribed test steps. Each test case is performed, step by step, and each deviation from expected behavior is recorded. The test plan was constructed to mirror the requirements such that successful completion of all test items ensures that requirements are met. A completed requirements traceability matrix confirms the testing coverage of the requirements by identifying which test cases cover which requirements.

3.2.6 Project Postmortem

The project postmortem includes compiling the team effort time logs, and adding the analysis in an updated software development plan. The complete listing of all meeting records is also added to the SDP, which is then re-baselined as a new version and made available. A user manual is created to explain the setup and operation of the system, and a project summary is written to provide an overview of the project from an academic summary perspective. The summary is also available as an html document for a web viewing, and has links to all other project artifacts ass

described. The code is counted and analyzed for complexity, which is then added to the code document. Finally, a presentation describing the project is created, and performed by Team Blue in a semi-formal dissertation.

4. Lessons Learned

- Prototyping is invaluable – it helped to get rid of a lot of low-level issues and basic functionality; just don't use the prototype as a product.
- The design document was very helpful, and drove the development, but parts were identified as being too much overhead - doing a quick throwaway prototype of the design may be useful.
- Technical issues can use up too much time - try to identify these large blocking ones in the beginning of the project.
- Pair coding was very valuable, especially when there are parts that need to be integrated.
- Integration can be flawless, provided the interfaces are agreed upon and created ahead of time!
- Team meet notes are good for documenting assumptions and thoughts for later follow up. This was essential during the requirements phase when discussing the system behavior and dynamics of operation.
- Webserver is has a mysterious and unresolved idiosyncrasy requiring that the image size be of a certain size to work correctly.
- Timing issues associated with real-time software development can result in unexpected behavior that are hard to debug if not designed for appropriately.