IPPD Training Manual

October 22, 2015

Table of Contents

1. Introduction
2. Overview of IPPD
3. IPPD Deliverables
4. IPPD Software Deliverables
5. Peer, Faculty and Sponsor Reviews
6. Expectations for Teams and Individuals
7. Evaluation Policy
8. Classroom Labs
9. Financial Policies and Procedures
10. Appendix

UF Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

Introduction

Welcome to the Integrated Product and Process Design Program; we are delighted to have you on board as a new engineer. We are excited to have you work with your teammates and an experienced engineer who will *coach* your team to complete a project sponsored by a company. This manual has been written to assist you in completing the project successfully and to teach you the Integrated Product and Process Design methods you will use throughout your career. It will also introduce you to some of the basic processes and policies of our corporation. Over an eight-month training program, new engineers are taught a structured design process and will get a chance to practice it in solving the customer's problem.

The training program provides both classroom and laboratory experience. You will learn:

- How fundamental engineering science is relevant to effective product and process design
- That design involves not just product function but also producibility, cost, schedule, reliability, customer preference and life cycle issues
- How to complete projects on time and within budget
- That engineering is a multidisciplinary effort

Working in small multidisciplinary project teams, you will get important practical experience in teamwork and communication and in developing your leadership, management and people skills.

Table of Contents

Introduction 2

Table of Contents 3

Overview of Integrated Product and Process Design 9

- IPPD program logistics 9
  - Canvas 9
  - How to Upload Assignments 9
  - Project activities 9
- Team Project Types (Hardware, Software, Combination, or Process) 11
- Hardware projects 12
  - Overview 12
  - Phase I - Conceptual design 12
  - Phase II - System level design 13
  - Phase III - Detail product and process design 13
  - Phase IV - Verification 14
  - Phase V - Production 14
  - IPPD Structured Design Process Chart (Hardware) 15
- Software projects 16
  - Overview 16
  - Phase I - System requirements and product design 16
  - Phase II - Detail design and comprehensive test planning 17
  - Phase III - Code/unit test/build and integration 18
  - Phase IV - Product verification 19
  - IPPD Structured Design Process Chart (Software) 21

IPPD Deliverables 22

- Introduction 22
  - Overview 22
  - Deliverable types 22
  - Packaging deliverables 23
- Technical strategy 24
  - Overview 24

Presenting this deliverable 25
Product design specification 25
Requirements 25
Specifications 29
Mapping specifications into requirements 30
Presenting this deliverable 34
Project roadmap 34
Overview 34
Presenting this deliverable 35
Concept generation/evaluation and selection 36
Concept generation 36
Functional decomposition 37
Functional decomposition 40
Concept generation for software 43
Presenting this deliverable 44
Concept selection 44
System/product architecture 46
Overview 46
Presenting this deliverable 47
Configuration management plan 47
Project plan 48
Overview 48
Project risks 49
Plan example 50
Presenting this deliverable 50
Business case 50
Overview 50
Presenting this deliverable 51
Prototype plan 51
Overview 51
Presenting this deliverable 53
Preliminary design report and project plan 53
Overview 53

Proposed content and presentation outline 53
Preliminary Design Report Peer Review 56
Product architecture 56
Overview 57
Create the schematic 57
Cluster the elements 59
Create a rough layout 62
Identify the interactions 63
Modularity leads to rapid progress 64
Presenting this deliverable 64
Component design specifications 65
Overview 65
Presenting this deliverable 66
Analytical and experimental plans/prototype test plan 66
Overview 66
Presenting this deliverable 67
Preliminary manufacturing plan/product cost 67
Overview 67
Presenting this deliverable 68
System level design report 69
Overview 69
SLDR Report submission requirements 72
Review One-on-One with Dr. Stanfill 73
Detail product and process design 74
Analytical and experimental report 74
Overview 74
Proposed report content 74
Presenting this deliverable 75
Detailed manufacturing process and tool documentation 75
Overview 75
Presenting this deliverable 76
Acceptance test results and report 76
Overview 76

Proposed report content 76
Presenting this deliverable 76
Final manufacturing and test plan 77
Overview 77
Presenting this deliverable 77
Final product cost 77
Overview 77
Presenting this deliverable 77
Final report and project documentation 77
Overview 77
Documentation requirements 78
Final Design Report 78
Due Dates 79
Meet with Dr. Stanfill 80
Signature page information 80
Submitting the report 80
IPPD Software Deliverables 85
Introduction 85
Overview 85
~~Technical strategy~~ 85
~~Overview~~ 85
~~Presenting this deliverable~~ 85
~~System/product requirements~~ 86
~~Overview~~ 86
~~Presenting this deliverable~~ 86
~~Complete and testable product specifications~~ 87
~~Overview~~ 87
~~Presenting this deliverable~~ 88
~~System/product architecture~~ 88
~~Overview~~ 88
~~Presenting this deliverable~~ 89
~~Prototype plans~~ 89
~~Overview~~ 89

~~Presenting this deliverable~~ 90
Preliminary design report 90
Comprehensive test planning 90
Comprehensive test plan 90
Unit test plan 91
Functional verification test plan 91
Product verification test plan 91
System verification test plan 92
Presenting this deliverable 92
Software development methodologies 92
~~Configuration management plan~~ 92
Reviews, inspections and walkthroughs 93
In-process measurements 93
Peer, Faculty, and Sponsor Reviews 95
Overview 95
Peer reviews with other teams 96
Preliminary Design Peer Review 96
System Level Design Peer Review 96
Final Design Peer Review 96
Reviews with sponsor companies 96
Preliminary Design Review 96
System Level Design Review 98
Final Design Review 98
Qualification Review Boards 98
Project Plan Review Session 98
Prototype Review Session 99
Expectations for Teams and Individuals 101
Team expectations 101
Overview 101
Team meetings 101
Team selection 101
Individual expectations 102
Overview 102

Design notebook 102
Team member reports 102
Individual lecture/workshop expectations 103
Evaluation Policy 104
Guidelines 104
Classroom Labs 105
Financial Policies and Procedures 106
Appendix: Weekly Status Memo 107
Appendix: Project Roadmap 108
Background and motivation 108
Proposed deliverable: the project roadmap 109
New section for the final report: process assessment 109
Appendix: Project Video Requirements 111
(See Prototype Demonstration Video section.) 111
Thoughts 111
Video capture 111
Video editing tools 111
Software tutorial tools 112

Overview of Integrated Product and Process Design

The Integrated Product and Process Design (IPPD) program is an eight-month training program for new engineers. In this program, new engineers from various disciplines are divided into five- or six-person teams. These teams work with an experienced engineer (team coach) and a liaison engineer from the customer to design and build real-life industrial projects. This chapter describes the overall IPPD program and the structured, top-down hardware and software development processes used to successfully complete projects on time and within budget.

IPPD program logistics

Canvas

Throughout the program, you will be using the IPPD site on Canvas to find out your weekly schedule, deliverables, announcements, and to upload your assignments. You will receive important email updates regularly from the Canvas site, so please be sure to read these emails. Also, keep in mind that IPPD *does read* everything you upload to the Canvas site.

Be sure to upload everything in deliverables, *even if they are still in draft mode.* IPPD wants to see where you are in the project. We realize it's not your final version of it, so we don't expect it to be perfect. In the professional world, we don't always have things completed and perfected when we turn them in. We hand them in, so we can discuss with the team and get feedback.

How to Upload Assignments

All deliverables must be uploaded to the IPPD Canvas site. For individual deliverables, respond to the instructions in the assignment directions. **Unless otherwise specified, all uploads are due by noon of the day of class.**

All teams are required to use Git as their project data repository.

To find more details on Canvas, Git, and the Team Wiki, *please consult your IPPD Professional Manual, which can be found on the course home page in Canvas.*

Project activities

Class lectures & workshops. During the eight-month period, engineering teams and their coaches will attend weekly lectures and workshops. There could be up to five lectures per week for all team members and coaches and at least one separate workshop for each team and their coach. During the lectures, new engineers will be introduced to major topics of a product and process design process. They will learn how to prepare for project deliverables that are generally due in the ensuing weeks. In the weekly team specific workshops, the teams and their coaches will customize the more generic lecture content to their specific projects.

Weekly team meetings. In addition to the weekly lectures and workshops, teams will meet separately to discuss status and assignments on their projects. Effective meeting practices include the creation of meeting minutes following each team meeting. These minutes should be distributed to your teammates, liaison engineer, and project coach. To keep the IPPD Director and project liaison engineer up to date on all project activities each week, **Weekly Status Memos** will be distributed. For consistency across all project teams, a web-based tool will be used to create and publish the Weekly Status Memos

Projects are assigned to coaches before the start of the training; this will allow the coach to visit with the sponsor company, to develop a better understanding of the project and establish a working relationship with the liaison engineer. Communication with the liaison engineer is expected during the weekly team meetings.

Weekly reviews with liaison engineers. During the eight months of the program, the engineering team will review the status of their project regularly with the liaison engineer. These reviews should be weekly at a predetermined time and need to be coordinated with the liaison engineer. Regular liaison communications reduces project risks and helps avoid surprises at major design reviews.

Design reviews. At a minimum, the project team will participate in three major design reviews with the sponsor company. The preliminary design review is held at the sponsor company's facility, while the system level and final design reviews are held on the University of Florida campus. Additional formal reviews with the sponsor may be scheduled if required to meet special project requirements.

Peer reviews. Prior to the three major design reviews with the sponsor company, a presentation practice session will be held with other project teams and coaches. The purpose of a peer review is to improve the content and delivery of important design information prior to presenting live in front of the project sponsor. The peer review audience evaluates each presentation and provides written feedback to the presenters. Peer reviews are typically scheduled for two consecutive periods. Attendance is mandatory.

Prototype inspections. Early prototyping is a key to success for engineered projects. Two prototype inspection sessions will be held. The first will occur in the fall semester two weeks ahead of the System Level Design Review. The second will occur at least three weeks ahead of the final design review. The format of the inspections will include a live demonstration of the project to an audience of faculty and staff members. The format varies from "speed dating" where the prototype is demonstrated to one or two new reviewers every 5 to 10 minutes over the course of an hour, or a single 5 to 10 demonstration to a larger audience.

Qualification Review Boards. The Qualification Review Board (QRB) is a group of faculty coaches that will attend your peer review sessions and will lead project reviews intended to expose project risks and provide recommendations for corrective action. The QRB will also review your major design reports for content. During the QRB sessions, each project team presents privately before the QRB subject matter experts. The sessions last thirty minutes and are scheduled during class time.

Design system training. Design system training may be provided in the form of specially scheduled workshops, computer-based training, or paper-based tutorials.

Team Project Types (Hardware, Software, Combination, or Process)

The following list gives you information about your team project and the project type (Hardware: HW, Software: SW, a combination of both, or Process: PR). Deliverables can differ, depending on your project type. Please check the list below to be sure you know what your project type is, so you will know what deliverables your team needs to upload throughout the year.

No.	Team	Project Type	Title
1	Just Wing It	HW	CAE KingAir
2	FiberFlex	PR	Optimize fiber reinforcement type, quantity, mixing, and forming to enhance the mechanical properties of CertainTeed Gypsum's wallboard
3	Florida Filtration	PR	Optimization of Waste Water Effluent
4	ImpactAid	HW/SW	Device to quantify bony fixation of orthopaedic implants in real-time during surgery
5	Apollo Connection	HW/SW	High-Speed Data Link
6	ZOOM	HW/SW	Engineering design unit (EDU) SRAM memory multi-module test station
7	StratuSstream	SW	Product Pulse Dashboard
8	SPOTR	HW/SW	Heterogeneous Robots Equipped for Search and Recovery
9	CerebroStim	SW	Integration of Wearable Sensors to Medtronic Nexus-D for Closed-Loop Deep Brain Stimulation
10	Cell Seeker	HW/PR	Modular, Disposable Packed-column Cell Separator
11	Rock Solid Solutions	PR	Acid Resistant Concrete Materials Design
12	Anti-Caking Engineering (ACE)	PR	Product and Process Factor Evaluation Affecting Quality of Ammonium Phosphate Fertilizers
13	Reverse Air	HW	3D Measurement, Modeling, and Fabrication for Aerospace Applications
14	AutoMed Solutions	HW	Automation of Bone Paste Syringe Filling
15	Drill Data Diagnostics	SW	Wireless Diagnostics For Mobile Drill Rigs
16	FERIT	HW/SW	Imaging Sensor Miniaturization for Gas Turbine Inspection
17	Crash Capture	SW	Crash Analyzer Portal
18	MouseTrap	SW	IT Security Communications Mobile Application
19	Guarden Technologies	HW/SW	Smart Garden/Pond Protector

Hardware projects

Overview

The hardware development process for the IPPD program is defined in five major phases:

1. **Phase I** Conceptual design
2. **Phase II** System level design
3. **Phase III** Detailed product and process design
4. **Phase IV** Verification
5. **Phase V** Production

In the following, each of these development phases are described as if they were standalone and sequential. In practice, these phases typically overlap and the content of each as described may have to be customized for different products.

The outcome of each development phase must be validated, verified and refined as necessary to eliminate as many problems as possible. Feedback into other phases may be required. If the product version at that design stage is satisfactory it should be baselined i.e. put under formal change control (configuration management).

The described hardware development process does not imply that a product as a whole has to move from one phase to the next. It may be effective to break a product into discrete modules and move them separately through the various phases. This could facilitate testing and rapid prototyping could provide early end user feedback.

While the content of each described development phase may be different depending on the product, dates for deliverables and major milestones do not change.

Phase I - Conceptual design

The major objectives of this development phase are the following:

- determine and document customer needs and product requirements = (HOQ)
- generate and evaluate alternate product concepts
- select one or more concepts for further development
- develop a set of specifications and benchmark competitive products
- produce an economic justification and development plan

The major tasks during Phase I are to develop the following:

- system/product requirements
- product design specifications
- concept generation, evaluation and selection
- a project plan
- a business case

Major development milestone	Timing
Preliminary design review with project sponsor	early October

Phase II - System level design

The system level design defines the product architecture and the decomposition of the product into subsystems and components (Ulrich and Eppinger: Product Design and Development, 2000). It embodies a detailed technical specification of the product to this point. This description includes overall design objectives for hardware and software, a description of the major subassemblies, components and software subsystems. The system level design provides proof that the selected design concepts have been verified will meet the product design specifications and can be translated into the detail design.

The focus in the system level design phase is on development of a product architecture that facilitates modularized development—allowing individuals or sub-groups from within the team to work in parallel. This allows the team to make much more rapid progress than the earlier serial tasks completed in the conceptual design phase.

The following major tasks must be completed during Phase II of the Hardware Development Process:

- product architecture
- component design specifications
- analytical and experimental plans
- proof of concept prototype
- prototype test plan
- manufacturing plan (project dependent)
- product cost (project dependent)

Major development milestone	Timing
Prototype inspection	late November
System level design review with project sponsor	early December

Phase III - Detail product and process design

The detail product and process design translates the system specifications into a structured set of detailed specifications of the product geometry, materials, and tolerances of all the unique parts in the product and the identification of all standard parts to be purchased from suppliers (Ulrich and Eppinger: Product Design and Development, 2000). These control documents and specifications include the following:

- part and assembly drawings
- printed circuit board layouts
- complete bill of materials
- software programs
- manufacturing process descriptions
- vendor part information
- assembly and fabrication process plans

- tooling drawings
- test procedures
- proposed tool and test equipment
- installation/maintenance procedures.

The following major tasks must be completed during Phase III of the Hardware Development Process:

- project plan and detail design review
- translate verified circuit simulations into routed printed circuit board layout
- develop accurate solid models of assembly and produce detail drawings of components and subsystems
- analytical and experimental report
- procure/fabricate functional prototypes
- develop detailed time lines for testing at module and integration levels
- update product specifications
- manufacturability assessment (project dependent)
- detail manufacturing and tool documentation (project dependent)

Major development milestone	Timing
Project plan review QRB	mid January
Prototype results QRB	late February

Phase IV - Verification

During the Verification phase, the final system prototype is tested against the product design specifications.

The following major tasks must be completed during Phase IV of the Hardware Development Process:

- manufacturing and test plan (project dependent)
- final product cost (project dependent)
- acceptance test results and report
- final report and documentation
- project poster
- project video

Major development milestone	Timing
Prototype inspection	late March
Final design review with project sponsor	mid April

Phase V - Production

The Production phase is listed here for completeness. There is not adequate time in the 8-month development process to address these activities. It is up to the project sponsor to initiate this activity.

IPPD Structured Design Process Chart (Hardware)

Semester One — Semester Two — Handoff to sponsor

▲PDR ▲SLDR △QRB1 △QRB2 ▲FDR

△ Prototype inspection △ Prototype inspection

Major tasks by phase

• system/product requirements • product design specifications • concept generation, evaluation and selection • project plan • business case	• product architecture • component design specifications • analytical and experimental plans • proof of concept • prototype test plan • manufacturing plan • product cost • create minimum viable product prototype	• project plan and detail design review (QRB1) • translate verified simulations into detail models/ drawings of components and subsystems • analytical and experimental report • procure/fabricate functional proto-types • prototype results & report (QRB2) • develop detailed time lines for testing at module and integration levels • update product specifications • manufacturability assessment • detail manufac-turing and tool documentation	• manufacturing and test plan • final product cost • fabricate final prototype • acceptance test results and report • final report and documentation • project poster • project video	• sponsor initiated activities

Key to symbols and abbreviations:
▲ = Major design report and review (sponsor approval)
△ = Internal faculty review
PDR = Preliminary Design Report
SLDR = System Level Design Report
FDR = Final Design Report;
QRB = Qualification Review Board

Software projects

Overview

The software development process for the IPPD program, created to manage the development of highly structured, embedded systems, is defined in four major phases:

1. **Phase I** System requirements and product design
2. **Phase II** Detailed design and test planning
3. **Phase III** Code/unit test/build and integration
4. **Phase IV** Product verification

The process may be used as the default development process for software projects, including application software and web-based tools[1]. In the following each of these development phases are described as if they were stand alone and sequential. In practice these phases typically overlap and the content of each as described may have to be customized for different products.

The outcome of each development phase must be validated, verified and refined as necessary to eliminate as many problems as possible. Feedback into other phases may be required. If the product version at that design stage is satisfactory it should be baselined i.e. put under formal change control (configuration management).

The described software development process does not imply that a product as a whole has to move from one phase to the next. It may be effective to break a product into increments of functional capability and move them separately through the various phases. This could facilitate testing and rapid prototyping could provide early feedback of user experience.

While the content of each described development phase may be different depending on the product, dates for deliverables and major milestones do not change.

Phase I - System requirements and product design

The major objectives of this development phase are the following:

- determine unambiguous, validated and prioritized requirements for the functions, interfaces and expected performance of the software product
- determine an unambiguous, verified set of specifications of the overall hardware-software architecture, control structure and data structure
- develop a preliminary design for each subsystem
- validate and verify requirements and specifications

[1] Recent software engineering practices for the development of application software are focusing on so-called "agile" methodologies. SCRUM is popular agile approach that has been effectively utilized on IPPD software projects since 2009. In lieu of following the 4-phase process defined in the following, project teams are welcome to utilize SCRUM to complete their development—provided that key deliverables are produced in concert with the default IPPD software process.

Major tasks
The major tasks during Phase I are to develop the following:

- a technical strategy
- system/product requirements
- complete and testable product specifications
- a system/product architecture, data flow diagrams, data modeling and class hierarchy
- preliminary plans for code reuse
- a preliminary design of each subsystem
- simulation and prototyping plans
- preliminary verification test plans
- configuration management control plans
- a project plan
- a business case

Major development milestone	Timing
Preliminary design review with project sponsor	early October

Phase II - Detail design and comprehensive test planning

The Detail Design translates the system specifications into a structured set of software components. Depending on the complexity of the product there may be up to 3 design stages in the programming architecture. The following defines each stage:

Product level design defines product structure hierarchy of classes, objects and system interfaces. In this design stage software components may be identified including function, data and interfaces.

Component level design defines component structure, subclasses and objects, modules, data definition and interfaces.

Module level design defines function, logic, data and interfaces for the smallest logical unit-usually less than 100 source instructions.

The detail design and comprehensive test planning of each of these stages must be captured in a Technical Design Specification that defines what actions must be performed to satisfy the system specification. To evaluate design alternatives and proof design concepts, rapid prototyping must be considered. As part of the detail design strong consideration must be given to maximize reuse of existing code. Comprehensive test planning and development of test cases must be done in parallel with the development of the detail design. A key objective of test planning is to provide enough details from the testing process to assess the overall design. Test planning should cover the following topics:

- code inspection methodologies
- unit and integration test
- functional verification test
- product verification test
- system verification test

- test environment
- test tools & equipment
- test simulators and drivers
- test stations/configurations needed

As with detail design, test planning should consider "reuse" opportunities including test plans and especially test cases. See the Preliminary design report

An outline for the

Preliminary design report and project plan major deliverable is provided in the

Comprehensive test planning section for additional details.

Major tasks
The following major tasks must be completed during Phase II of the Software Development Process:

- detail module/component and product level designs have been completed and verified
- a comprehensive test plan has been established
- the detail designs and the comprehensive test plan are captured in the "Technical Design Specifications"
- configuration management control has been implemented.

Major development milestone	Timing
Technical design specification review with project sponsor	mid November
Prototype inspection	late November
System level design review with project sponsor	early December

Phase III - Code/unit test/build and integration

The major objective of this development phase is to translate the design details into source code. Code inspection and unit test for the smallest logical software module follow this. Unit test is a repetitive process and continues until all modules have been thoroughly tested. Typically test cases during Unit Test deal with the following:

- interface testing
- branch coverage
- boundary testing
- error cases
- output generation

Code and Unit test is followed by build and integration. The purpose of the build process is to transform all developed source code into executable software. All modified/created source code modules, programmable information and all reused code modules will be compiled for unique identification, control and management. The integration test verifies that the functionality and stability of the developed software product is maintained when all the features are integrated into a single product build.

Major tasks
The following major tasks must be completed during Phase III of the Software Development Process:

- source code has been created and inspected
- all new and changed code must be successfully unit tested
- the source code must be integrated into a change controlled database (configuration management control)
- the comprehensive test plan should be refined
- test cases for functional, product and system verification tests have been developed.

Major development milestone	Timing
Project plan review QRB	mid January
Prototype results QRB	late February
Test readiness review with sponsor	late February

Phase IV - Product verification

Product verification generally encompasses three formal test stages following unit testing-functional verification test, product verification test and system verification test/acceptance test.

Functional verification test

This test verifies proper execution of product components and does not require that the product being tested interface with other program products. This could allow testing of a product before others are ready. It may also be more effective to initially test in a controlled and isolated environment.

Major tasks
Verify successful execution for all function, module and component interfaces and operational characteristics against final programming specifications. Test different values for parameters and variables. Execute all new and modified function paths. Recommend actions as needed.

Product verification test

This test verifies the proper execution of the entire product against defined product specifications. It requires a direct interface with other program products.

Major tasks
Verify successful execution for all functions, operational characteristics and interfaces between modules, components and other system products against final programming specifications. Execute all new and modified functions and error paths. Recommend actions as needed.

- acceptance test results and report
- final report and documentation
- project poster
- project video

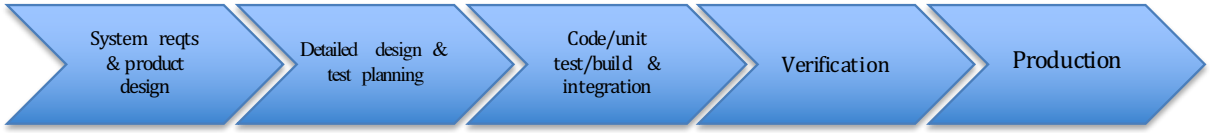
System verification test/acceptance test

This test verifies proper execution of all related software and hardware products and is performed using only the actual software and hardware.

Major tasks
Verify execution of the software product with all related software and hardware products, using only the actual software and hardware applicable against final programming specifications

Major development milestone	Timing
Prototype inspection	late March
Final design review with project sponsor	mid April

IPPD Structured Design Process Chart (Software)



Semester One — Semester Two — Handoff to sponsor

▲PDR ▲SLDR △QRB1 △QRB2 ▲FDR

△ Prototype inspection △ Prototype inspection

Major tasks by phase

• technical strategy • system/product requirements • complete and testable product specifications • system/product architecture, data flow diagrams, data modeling and class hierarchy • preliminary plans for code reuse • preliminary design of each subsystem • simulation and prototyping plans • preliminary verification test plans • configuration management control plans • project plan • business case	• detail module/ component and product level designs have been completed and verified • comprehensive test plan established • detail designs and the comprehensive test plan are captured in the "Technical Design Specifications" report (SLDR) • configuration management control has been implemented • create minimum viable product prototype	• project plan and detail design review (QRB1) • source code has been created and inspected • all new and changed code successfully unit tested • the source code integrated into a change controlled database (configuration management control) • comprehensive test plan refined • prototype results & report (QRB2) • test cases for functional, product and system verification tests have been developed.	• complete final prototype • acceptance test results and report • final report and documentation • project poster • project video	• sponsor initiated activities

Key to symbols and abbreviations:
▲ = Major design report and review (sponsor approval)
△ = Internal faculty review
PDR = Preliminary Design Report
SLDR = System Level Design Report
FDR = Final Design Report;
QRB = Qualification Review Board

IPPD Deliverables

Introduction

The progress of engineering design projects is generally managed through a phase review process. Targets are established for each of the phases. Progress is determined based on achieving these targets at cost and on time as determined in the overall project plan. To achieve a target, the design team must deliver a pre-specified target content. Major targets in the design are therefore marked by deliverables.

Overview

In the Integrated Product and Process Design program we will choose a set of deliverables that are generally applicable to engineering design projects. They will allow us to monitor the progress of a project, to evaluate the student team and to report the status of the project to the industry sponsor. Deliverables that are specific to a particular type of project content are labeled in this chapter with the following designations:

Label	Project type	Examples
Hardware	hardware oriented	electro-mechanical system development, test and measurement
Software	software-oriented	mobile apps, embedded systems, decision support tools, real-time software
Process	process oriented	optimization of process control characteristics, quick changeover manufacturing process design, quality improvement of process streams

Deliverables that are not otherwise labeled should be assumed to be applicable to all projects.

As discussed in the Software projects section, agile development methodologies, such as SCRUM, may be used in lieu of the structured development process described in this chapter. Your project coach can assist you in determining how to adapt the content and timing of the major software deliverables described in the following sections to your particular project.

Deliverable types

In the IPPD program we will refer to the following three types of deliverables:

1. **major deliverables** that are required of all projects; these include the **Preliminary Design Report**, the **System Level Design Report**, the **Final Design Report**, and the respective design report presentations, the project poster, and the final prototype system. Milestones and deadlines associated with the major deliverables may not be changed.

2. **intermediate deliverables** such as the elements defined in this chapter that are project dependent. The content and sequence of intermediate deliverables are not critical provided appropriate content is included in the major deliverables.
3. **course deliverables** such as team, individual, class, and coach evaluations; **weekly status memos**; meeting minutes; pre- and post self-assessments; travel requests and materials and supplies requests; checklists for design reviews and end-of-semester requirements.

Intermediate deliverables are almost always appropriate. In certain circumstances, detailed deliverables may not be appropriate for a given project. Faculty coaches and liaison engineers would have to work out the specifics of what seems more appropriate for a particular project. The project roadmap provides a mechanism to document planned process deviations from the standard IPPD process—such as delaying or eliminating a particular intermediate deliverable. For example, a project that requires significant experimentation may delay concept selection until after the scheduled delivery of the Preliminary Design Report. Another project may not result in a manufactured product, so the Product Cost and Manufacturing Plan intermediate deliverables may not make sense. Note that specific deliverable content may be changed if absolutely necessary—for instance, the *Product Cost* deliverable may be modified to be a *Process Cost* deliverable.

Packaging deliverables

Deliverables consist of a concise written report with supporting project data (charts, pictures, graphs, analyses, tables, spreadsheets, research data) from which a short PowerPoint presentation is derived. The PowerPoint presentation must be ready to be presented during class on the week it is due. Prior to presenting the deliverable during lecture to peers, the content should be reviewed and approved by the faculty coach and liaison engineer. While all IPPD student engineers, staff and faculty have signed off on a Non-disclosure Agreement[2], it is still prudent to avoid presenting proprietary sponsor information unless cleared to do so by the liaison engineer.

Naming conventions: *<team name>-<deliverable name>.<type>*[3]
Cybertexters-PDS.pptx
Cybertexters-ProdCost.xlsx
Cybertexters-ProdArch.docx

Upload to Git: All work-in-process and finalized deliverable content in all digital formats should be accessible on the project team's Git repository. All team members are expected to utilize the repository. The repository should be organized so that the liaison engineer, coach or IPPD staff members can locate and review your work. Deliverables should be placed in the *deliverables* folder in the team's Git repository.

Upload to the team's folder in Canvas: Rather than uploading deliverable documents to Canvas, the deliverable assignment shall be satisfied by submitting the appropriate links (URLs) to the Canvas assignment. By noon of the class meeting day, one team member shall ensure that the deliverable and its associated files are uploaded to the team Git repository and that the links

[2] "Assignment of Ownership and Nondisclosure Agreement," found on my.ippd.ufl.edu
[3] Including the revision number in the filename is not recommended for files managed in a Git repository

to these files are included in the Canvas assignment submission. Be prepared to present your deliverable to the entire IPPD audience at the start of class. Revision of the deliverables are permissible beyond the stated due date and time.

Available on team wiki: The wiki is tightly integrated with Git and can serve as a useful tool to organize the access to specific documents in the repository, such as reports, deliverables, and meeting minutes.

Time savers: Deliverables documented in both a brief report format and a PowerPoint format are easy to consolidate into first drafts for the major reports and design presentations.

Develop reusable introduction slides: To ensure consistency in presenting deliverables to peers and sponsor company engineers, a set of suggested presentation guidelines accompany the deliverable descriptions that follow. *Note that these are guidelines and may be deviated from as needed to satisfy the intent of the project—do not prepare or present content that does not support the project outcomes*. If you have questions, consult your coach or the Director (in that order).

Further, it is suggested that each development team create 1 to 2 presentation slides that provide the following project information:

- team name and logo
- project title
- sponsor company
- team members--including the faculty coach and liaison engineers
- a concise description of the project with key objectives

Elevator speech: It is suggested that the team develop an elevator speech to describe the project. The description should be about 75 to 140 words (one minute maximum). Each team member should be able to repeat the script, or their own variation, from memory. This script should be used to introduce the project for all design reviews and deliverable presentations. Be sure everyone on the project team can give the elevator speech.

The following sections describe the major project deliverables.

Technical strategy

Software

Overview

Requirements for a product must be evaluated in the context of a technical strategy that provides the strategic direction for the product. The technical strategy should describe:

- the basic structure and direction of the product
- the development methodology (i.e. SCRUM)

- the relationship of the product to any pertinent architectural standards[4]
- user interface standards
- interface standards with other products
- direction for modification or replacement of old source code
- the software engineering environment and tools required for the product such as:
    - programming language
    - compilers
    - equipment and operating systems

The technical strategy does not describe the internal design or implementation of the product.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides with basic structure & product direction; a list of pertinent architectural, user interface standards, and system interface standards
- 1 slide with targets for reuse of existing requirements, specifications and code
- 1 slide with list of software environment, tools, programming environment, compliers, equipment & OS (mention configuration management system if known)
- 1 slide with basic development methodology plan (for instance, if you plan to use SCRUM, how often will sprints be scheduled, what units will you use for estimation—days, half-days, hours, etc., how will the burn-down chart be managed, etc.)

Product design specification

Hardware **Software** **Process**

The Product Design Specification[5] (PDS) is the technical response to customer wants and needs. The PDS captures the voice of the customer and translates the customer's qualitative expectations of the product or process (requirements) into into quantitative targets (specifications) that can measured. The PDS must be comprehensive and unambiguous for both hardware and software components of the design. It must provide specifics about what is to be designed and built and how it will perform.

Metrics of the PDS should, if possible, be quantified and defined in the clearest possible terms. It is a formal statement of what a product has to do and is the fundamental control mechanism and basic reference source for the entire product development activity.

Suggestions pertinent for software-oriented PDS will be provided in addition to the broadly applicable PDS content.

Requirements

Determining requirements or customer needs for a product can be challenging because requirements are not always known with precision at the outset and also because they may

[4] These standards include, but are not limited to ANSI, ISO, ECMA, W3C, IETF, UML, CORBA, HTTP, PDF, Windows, IEEE 1471, DoDAF, C4ISR, TOGAF, RM-ODP, SysML, and MDA

[5] "Process" can be substituted for the word "Product" throughout this deliverable description

change over time. The objective of this part of the PDS deliverable is to determine requirements to be met by the design of the product. Requirements should be unambiguous, validated and prioritized by the customer of the product. Once requirements have been determined they need to be *baselined* i.e. stored in a requirements database. A spreadsheet can be used store the requirements list.

The Scope of Work (SOW) for a project is a rich source of content for developing project requirements. The liaison engineer (your customer/client), your project coach, and even your customer's customer (important to know this!) can be excellent sources of requirements. To get started, it is useful to generate a list of project stakeholders and their needs (we may also use the word "feature" to represent requirements). Figure 1 shows a graphical representation of stakeholders and associated requirements.

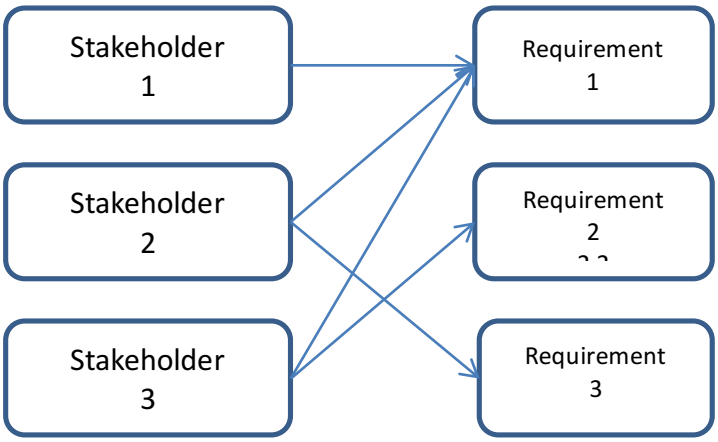


Figure 1 Graphical representation of stakeholders and associated requirements

Note that some requirements have more connections than others. These may be the most important needs/requirements to satisfy. Figure 4 shows an example[6] with more detail for a sample product.

[6] Adapted from ASEE Workshop: Helping Students Achieve a "Systems Perspective" of Engineering, June 14, 2015

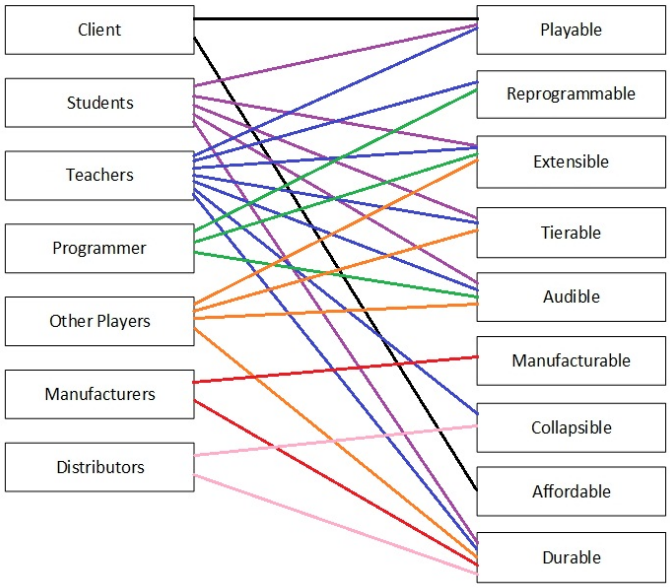


Figure 2 Stakeholders and associated requirements/features for a consumer product

Common stakeholders

The following are commonly used stakeholders[7]. The list is not exhaustive and is highly project-specific.

- end user
- client
- other engineers or scientists
- regulatory agencies
- those who maintain or repair or update
- societal groups (i.e. government, first responders, Millennials, Gen X, Gen Y, Baby Boomers)
- manufacturers, fabricators, assemblers
- shipping department
- marketing/sales/retail department
- legal department
- quality assurance
- product testing
- those responsible for disposal/deletion of software

Common features

The following are common project features[8]. The list is not exhaustive and is highly project-specific.

- something to describe the project's primary purpose
- affordable
- adaptable
- recyclable
- secure
- robust

[7] Adapted from ASEE Workshop: Helping Students Achieve a "Systems Perspective" of Engineering, June 14, 2015
[8] Adapted from ASEE Workshop: Helping Students Achieve a "Systems Perspective" of Engineering, June 14, 2015

- small-size/form factor/weight (compactness)
- easy to use
- safe to use
- simple
- efficient
- environmentally friendly
- reparable

Requirements differ from specifications

Requirements and specifications are not the same. Sometimes the term *requirement* and the term *specification* are used synonymously. IPPD development process distinguishes between the two. Requirements refer to qualitative descriptions of customer needs and wants that are written in the customer's language (understandable by a non-technical person). Specifications are quantitative, measurable embodiments of one or more requirements. Specifications are written in the engineer's language (very specific, with target metrics). Engineers apply a transformation to a set of requirements that results in a set of specifications. If all the specifications are met, then all the requirements should be met (and the customer is happy).

The examples that follow illustrate the differences between the requirements and specifications.

Customer needs/requirements examples:

<R-12>	The software shall provide real-time response to sales activity queries
<R-5>	The system shall be reliable
<R-33>	The housing must remain cool during operation

Note that these requirements do not provide a clear idea of how each would be proven. <R-12> may be adequate for determining programming objectives, but it is not precise enough to define a test to measure success. The more testable version of this requirement (the specification), could be:

Type A queries in less than .5 sec.

Type B queries in less than 1 sec.

Type C queries in less than 2 sec.

Software requirements

Software

The following four-step process may be useful for determining software requirements.

Collecting input requirements: Each input requirement is numbered and includes a brief description of attributes related to the intended function. Customers of the product are generally the source of input requirements. Market research, literature and competitive analysis may be other resources.

Problem analysis: This analysis should provide a detail description of each underlying problem from an end user prospective. It contains information needed to understand the requirement. In addition to control information it may contain a list of problem conditions, descriptions of current functions, user tasks, end-users and the environment. Analyzing problems can be an iterative process until it is agreed that the problem description is complete, consistent and correct.

Solution analysis & definition: The purpose of solution analysis and definition is to develop one or more effective, external solutions for each selected problem definition. It may include control information, proposed function, resolved problem conditions, modified or new user tasks, affected environment and end-user, and a value assessment for the particular solution.

Programming objectives: Programming objectives incorporate a set of solution definitions into a coordinated description of new and enhanced functions. As programming objectives are being developed the level of solutions as previously defined may have to be refined. This is an iterative process until the Programming Objectives satisfy the Problem Definitions.

System modeling can be a part of requirements definition. Entity relationship diagrams (ERD) would be one example of data modeling aspect of a system model. Viewpoint or data flow models might be another. For software projects where the user requirements are not well understood, it is recommended to follow an agile development process, such as SCRUM, where a series of minimally viable prototype applications are built rapidly and reviewed with the end users (customers) for feedback and redirection as needed.

Specifications

Hardware **Software** **Process**

Specifications must be testable to the extent that one can define a clear pass/fail test for determining whether or not the developed product will satisfy the specifications. In order to be testable, specifications must be specific, unambiguous and quantitative wherever possible. All specifications must be numbered—especially for reference in test documents.

Specification examples:

<S-16>	The product shall be capable of supporting data signaling transfer among modules in a full duplex rate from 56 Kbps to 4 Mbs.
<S-17>	The product shall support a time and date clock. Time and date shall be retained across power cycles.
<S-21>	The system shall operate continuously for 1000 hours without operator intervention
<S-3>	The housing temperature shall not exceed 30 deg C during operation

It may require significant effort to eliminate the vagueness and ambiguity in a specification and make it testable. It may require iteration (i.e. rapid prototyping) between requirements and specification to achieve affordable testability. Determining testable specifications at this stage is also important in preparation for a product verification strategy.

Software specifications

Software

Specifications must be testable to the extent that one can define a clear pass/fail test for determining whether or not the developed software will satisfy the specifications. In order to be testable, specifications must be specific, unambiguous and quantitative wherever possible. All specifications must be numbers--especially for reference in test documents.

Example: Specifications 16 and 17

<S-16>	The product shall be capable of supporting data signaling transfer among modules in a full duplex rate from 56 Kbps to 4 Mbs.
<S-17>	The product shall support a time and date clock. Time and date shall be retained across power cycles.

Software requirements and product specifications checklist:

1. Is the requirements document complete, i.e., does it reflect all of the known customer needs?
2. Are the specifications so detailed that designing and coding are restricted to a single implementation?
3. Does the human interface follow established standards?
4. Has the entire infrastructure been specified, i.e., backup, recovery, initialization, etc.
5. Have all reliability and performance issues been listed?
6. Have all security considerations been listed?
7. Do the requirements consider all existing constraints?
8. Do the requirements provide an adequate base for design?
9. Are the requirements complete, correct, and unambiguous?
10. Are all specifications testable?
11. Are all interface requirements feasible?

Software development should only proceed after satisfactory answers to the questions in this checklist have been provided or actions to resolve outstanding issues have been defined with a date for resolution.

Mapping specifications into requirements

Hardware Software Process

For completeness, every requirement should be *covered* by at least one specification. C*overed* means that the user need articulated in the requirement has at least one measurable specification associated with it. Table 1 shows a matrix representation of the specifications that cover each requirement.

Table 1 Requirements coverage matrix showing which specifications apply to a each requirement

No.	Requirement	Specifications covered
R-1	The system shall be safe to use	S-2, S-3, S-5, S-6
R-2	The user interface shall be easy to use	S-3, S-4, S-15
R-3	The system shall require little maintenance	S-1, S-2

Conversely, to show that every specification has at least one corresponding requirement, the reverse mapping can be presented. shows a matrix representation of the requirements that are covered by each specification.

Table 2 Specifications coverage matrix showing which requirements apply to each specification

No.	Specification	Requirements covered
S-1	The MTBF shall exceed 1000 hours	R-3
S-2	The system shall operate without lubrication for 2 hours	R-1, R-3
S-3	The system shall conform to the SAE J2929 standard	R-1, R-2

The relationship between the requirements and specifications can also be depicted graphically; however, this method can quickly become unwieldy.

House of Quality

Hardware **Software** **Process**

The House of Quality (HOQ) is another table-based approach for illustrating the relationship between requirements and specifications. The HOQ combines customer preferences and estimation of how well a given specification correlates with a given requirement to provide a priority ranking of the specifications. IPPD will use just a small portion of overall HOQ methodology and focus on identifying the most important specifications for initial focus.

Figure 3 shows an HOQ for a new technology steam iron. The simplified HOQ does not include a roof area above the specifications columns. The roof area provides a visual correlation between the specifications. Positively correlated attributes will each improve if either improve. For instance, an improvement in drag coefficient will simultaneously improve fuel consumption rate. Negatively correlated attributes will behave oppositely. For example, an improvement in acceleration may increase the fuel consumption rate.

				Specification													
	Category	Customer need	Imp.	S01 Cost <= $69 MSRP	S02 Heat-up time <= 30 sec	S03 Cool-down time <= 60 sec	S04 Steam jet - 0 sec. between bursts in vert. pos.	S05 No water fails to be vaporized in vert. pos.	S06 No water drips out of iron when turned off	S07 Function buttons with 1 psi pressure	S08 Temperature within 10 ° of target temp.	S09 Burst steam	S10 Burst spray	S11 Continuous steam	S12 Continuous spray	S13 Heat-up complete signal
PRODUCT	*Cost*	R01 Needs to be cost effective	4	9	3	3	3	3	3	3	3	3	3	3	3	3
	Features	R02 Instant heat	4	3	9	3			1		1					9
		R03 Instant cool	4	3	3	9			1		1					
		R04 Constant vertical steam	4	3			9	3	1			3		3		
		R05 Self adjusting to fabric	3	9												
		R06 Anti-drip	5	3				9	9			1		1		
		R07 Comfortable handle	1	9												
		R08 Burst & continuous steam	4	3	1	1	3		3	3	1	9		9		
		R09 Burst & continuous spray	4	3						3			9		9	
		R10 User interface	3	9			1		1	9		1	1	1	1	3
	Quality	R11 Tight temperature range	2	3	1	1		9	3		9					3
	Safety	R12 Auto off	4	3					1							
		R13 Safe temperature control placement	2	3												
		Weighted Column Totals (Product)		**198**	**66**	**66**	**63**	**87**	**94**	**63**	**42**	**68**	**51**	**68**	**51**	**63**
		Rank		**1**	**6**	**6**	**8**	**3**	**2**	**8**	**13**	**4**	**11**	**4**	**11**	**8**

IMPORTANCE RANGE 1 to 5
Correlation: 1=Low, 3=Medium, 9=High

Figure 3 Simplified House of Quality example from a new technology steam iron project

Each of the customer needs (requirements) has an importance rating from 1 to 5, where 1 is least important and 5 is most important. The specifications (columns) are correlated with the needs (rows) and a correlation assessment is applied. Low correlation is set to 1, medium to 3, and high to 9. Weighted totals for each column are used to establish the ranking of the most important specifications. The highest ranked specifications should be given the highest priority.

A note on the customer importance rating: it may be difficult for your customer to articulate the importance ratings—sometimes it seems like everything is important (therefore nothing is important). Asking the customer to make pairwise rankings can provide insights into overall rankings, though if there many comparisons to make, this will be time consuming and potentially irritating to your client. Another approach is to use the stakeholder-feature (needs) diagram and base the importance ratings on the number of arcs connecting a particular element.

Technical Performance Measures

Hardware **Software** **Process**

A Technical Performance Measure (TPM) is a critical specification that will be tracked for technical progress throughout the IPPD process. Three to seven specifications should be

designated as TPMs. Progress toward meeting TPMs will be reported during the following IPPD milestone activities: PDR, SLDR, QRB1, QRB2, and FDR. The progress is reported as the current level of achievement for the TPM.

The following are example TPMs:

- flow rate of 100 GPM sustained for 3 hours
- data rate of 25 Mbit/sec over 15 kilometers
- natural frequency of greater than 7 kHz
- target acquisition in less than 2 seconds

The following is an example of a TPM matrix reported at the FDR. Note the color scheme. Red indicates current TPM achievement is unacceptable. Yellow indicates trouble, while green indicates the target has been met. Blue indicates the target has been exceeded—in other words, the design team can demonstrate a positive margin for this TPM. The color abbreviations serve two purposes. First, a significant portion of the population is color blind. Second, the chart may be printed in black and white and the color lost. Note how the each TPM progresses during the IPPD project.

Technical Performance Measure (TPM)	Target	Actuals				
S01 MSRP	< $69	75.34 R	71.03 Y	70.15 Y	69.28 Y	68.92 G
S06 Water leakage rate when iron off	< 0.001 ml/min	1.1 R	0.51 R	0.12 R	0.010 R	0.001 G
S02 Heat-up time	< 30 sec.	33 Y	33 Y	31 Y	30 G	30 G
S03 Cool-down time	< 60 sec.	123 R	97 R	82 R	70 Y	59 G
S14 Steam jet burst delay	< 0.5 sec.	.5 G	.5 G	.48 G	.42 B	.42 B
	Milestones:	PDR	SLDR	QRB1	QRB2	FDR
Codes:	R(ed) = unacceptabl	Y(ellow) = in trouble	G(reen) = met	B(lue) = +10%		

Required PDS elements

Hardware **Software** **Process**

The PDS should consist of the following elements:

- a numbered list of requirements with priorities and rankings
- a numbered list of specifications with target metrics and direction of improvement (> target value, < target value, = target value, etc.)
- a matrix showing the mapping of the requirements into the specifications—all requirements should be covered by at least one specification; the House of Quality can be used for this representation
- a chart identifying the key specifications that will be tracked as Technical Performance Measures (TPM) during the project

The PDS should be an up-to-date working document as the design process proceeds; however, upon completion of the design, the PDS is a written document that matches the product itself. This must be verified through an acceptance test where the prototype or production sample is tested against the PDS. The PDS also serves as the basis for the contractual agreement between

the design team and the sponsor company. It should be presented to the sponsor company as part of the preliminary design review. This allows the sponsor company to make any objections known before the actual more detailed design work is started. Establishing product specifications and selecting design concepts is an integrated/iterative process.

Consider the following important points regarding the PDS:

1. The PDS is a control document. It represents the specification of what you want to achieve—not the achievement itself.
2. The PDS is a user document. Write succinctly and clearly.
3. The PDS is not an essay. Use short, sharp definitive statements.
4. The PDS should distinguish between customer NEEDS that must be met under all circumstances and customer WANTS that should be taken into consideration if possible.
5. The PDS metrics should be quantified. If in doubt estimate and verify later through experiments. (Remember you want to test the product against the PDS)
6. The PDS document and any modifications must be clearly dated.
7. The most critical specifications should be identified as Technical Performance Measures (TPM). Progress toward achieving each TPM will be reported at each major milestone in the IPPD process.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide with prioritized customer needs
- 1 slide with a list of specifications with target values
- 1 slide with a house of quality (HOQ) demonstrating the correlation between the specifications and the customer needs
- 1 slide summarizing the most important specifications (TPMs) as determined from the HOQ, organized by IPPD milestone

Project roadmap

Hardware **Software** **Process**

Overview

The project roadmap is a statement of process. The roadmap lists the IPPD deliverables the team will be completing and an overlay of any special sponsor deliverables or review requirements. The team will need to understand the IPPD deliverables early and work with the coach and sponsor to ascertain which deliverables are applicable and the appropriate timing. The project roadmap deliverable will be due within a week of the first sponsor visit. This early process-oriented effort will help the team better understand how the project will be tackled and provide early visibility into the detailed project plan.

Consider the following key information when creating the project roadmap (and ultimately when completing the deliverables outlined on the roadmap):

- before starting the project roadmap document, read and understand the descriptions for each deliverable. Discuss the deliverable requirements with your team members, coach and liaison engineer.

- follow the *spirit* of the deliverable/process element and not the *letter of the law* of each

- weigh carefully the implications for deviating from the "classic" IPPD process

- do not get bogged down in trying to create content for every deliverable if the deliverable does not make sense for the project. Negotiate the applicable aspects

- major deliverables, such as the Preliminary Design Report, the System Level Design Report, the Final Design Report, the project poster, the project video, and the final prototype are required for every project. Content of these project elements will vary depending on the project.

- intermediate deliverables, such as the elements defined in this chapter are project dependent. For example, **software specifications** may not be available until after significant prototype testing has been completed. In the roadmap, the team may split this deliverable into a preliminary product specification occurring prior to the Preliminary Design Report and the final product specification occurring prior to the System Level Design Report.

The following is a partial list of things that may be included in the project roadmap:

- time for feasibility analysis
- additional prototypes
- background research phase
- a hybrid deliverable process utilizing both hardware and software deliverables
- others

A template document for the roadmap will be provided by your coach. This document will be in Excel format. It is recommended to maintain the roadmap in this format.

The roadmap will be submitted to the IPPD Director and a **qualification review board** to approve or recommend modifications to the roadmap.

Final report process assessment: The **final report** will include an assessment of the process followed by the project team. This assessment will illustrate the actual roadmap followed, detail what worked and did not work, and will provide recommendations for future projects. The **project roadmap** will be made available for future project teams.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide per IPPD project phase illustrating the sequence of the intermediate deliverables and highlighting any sponsor-specific deliverables

Concept generation/evaluation and selection

Hardware Software Process

Concept generation

Completion of the product design specification (PDS) provides a better understanding of the design problem, constraints involved and final design objectives. With this understanding the design team can now generate, evaluate and select design/manufacturing concepts that will satisfy the product design specification.

Concept generation, evaluation and selection and the definition of product design specifications can be iterative in order to arrive at the best design concept. As you generate and evaluate concepts it may require clarification or modification of the PDS

- because of additional/new information about customer wants and needs
- to allow development of a solution in the form of a design·
- to meet the financial objectives.

This phase in the design process is called the conceptual design. It is completed over the first six to seven weeks of class. It forces a formalization of the product design to this point. It is the major part in the preliminary design report that will be reviewed by the sponsor company. It also provides valuable insight into how well the team understands the problem. Approval of the conceptual design is the basis for moving on to the next stage of the design process: The System Level Design.

Important points:

1. Try to get as many ideas as you can possibly generate.
2. Record ideas as they occur and number them.
3. Avoid early temptation to start engineering and developing ideas in detail and making firm decisions during concept generation.
4. Do not reinvent the wheel. Understand competitive or similar products and analyze them, look at analogous problem (or system) solutions, benchmark, research the market, do a thorough literature and patent search.
5. Do not reject brilliant concepts because they conflict with the PDS. Maybe the PDS has to be changed—remember the PDS is evolutionary.
6. Never proceed without a PDS.
7. Consider using concept generation tools, such as the TRIZ/TIPS process.
8. Engineer your concepts as completely and recognizable as possible. Stay within the laws of physics.
9. Try to assess the feasibility of concepts. As required use the experience of your liaison engineer or faculty coach.
10. Assess the technology required in your concepts for compatibility, completeness, availability and maturity.
11. Think about what parts or subsystems of the selected concept you probably would make or buy for your prototype. Discuss product/part sourcing with your sponsor company at the preliminary design review.

12. Think about how you would prove your concept and test your prototype. Discuss this with your sponsor company at the preliminary design review.

Functional decomposition

A key to success in concept generation is decompose the desired system into a structured representation. This representation may reflect a network of user interactions or perhaps a network of purely functional elements. A systems-based approach utilizes elements of user interactions and the functional elements to build a functional architecture. Once the key functional elements are identified, a series of structured and unstructured generative tools will be employed to create concepts.

Defining system actors

The first step is to revisit the list of stakeholders developed in the PDS, and identify those that will physically interact with the system. Next, determine what physical things will interact with the system. These physical things include, but are not limited to items such as power outlets, network cable connections, other machines or equipment, heat, fluids, foundations and floors, unintended devices and objects, and as a catch all for consumer-oriented devices, dogs and babies (think of Fido chewing on your smartphone or TV remote). Anything that interacts physically with the system will be referred to as an actor—this includes those stakeholders/users and the physical items mentioned above. Actors are nouns. Figure 4 shows an actor interacting with a system. The arc connecting the actor to the system is labeled as an Input/Output (I/O).

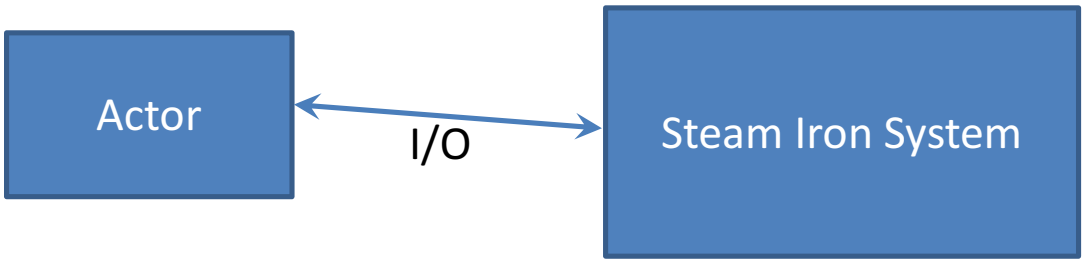


Figure 4 System model showing an actor interacting with a remote control system

Table 3 shows a list of actors with the associated descriptions. Note that not all of the actors are human. It also follows that not all actors are stakeholders. Stakeholders have a vested interest in the project outcomes. Dogs and babies would not be stakeholders unless projects are designed specifically for their use.

Table 3 Actors and associated descriptions for a digitally controlled steam iron system

Actor	Description
user	person trying to de-wrinkle fabrics
wall outlet	provides power for heating the sole plate and powering controls
fluids	liquids that may spill onto the iron; fluids that may fill up the reservoir
assembly workers	individuals who assemble and package the iron
ironing board	primary device to support fabrics while de-wrinkling
floor	landing zone for dropped iron

Determining inputs/outputs

Inputs/Outputs (I/Os) describe *what is being transferred* between the actors and the system. I/Os are nouns. Common examples of I/Os include the following:

- signals
- power
- energy
- force
- information
- mass
- light
- heat
- voltage
- current

Each of the I/Os should be described and attributed to the different actors previously defined. Table 4shows an example of a few I/Os from a sample project.

Table 4 Sample descriptions of I/Os for a digitally controlled steam iron

I/O	Description	Actors
Grip force	Force applied by user to hold and guide the iron	User
Heat up signal	Visual or audible signal that iron is ready	User
Power	Power to wall outlet from the electrical power grid	Wall power outlet Power grid
Plug-in/pull-out force	Force applied to insert or remove the power cord plug to/from the wall outlet	Wall power outlet User
Heat energy	Energy flow from the iron's soleplate to the fabric or atmosphere	Fabric Ironing board
Steam burst mass transfer	Flow of vaporized water (steam) from soleplate to fabric or environment	Environment Fabric Ironing board
Selection	Command applied to the UI to select iron settings	User
Impulse	Impact force of iron from a fall	User Dogs & babies Floor
Reaction force	Force of fabric and ironing board transmitted back to iron through the soleplate	User Fabric Ironing board

Interactions

Interactions describe what happens between the actors and the system. Interactions are verbs followed by nouns. Figure 5 shows a few interactions for a sample project.

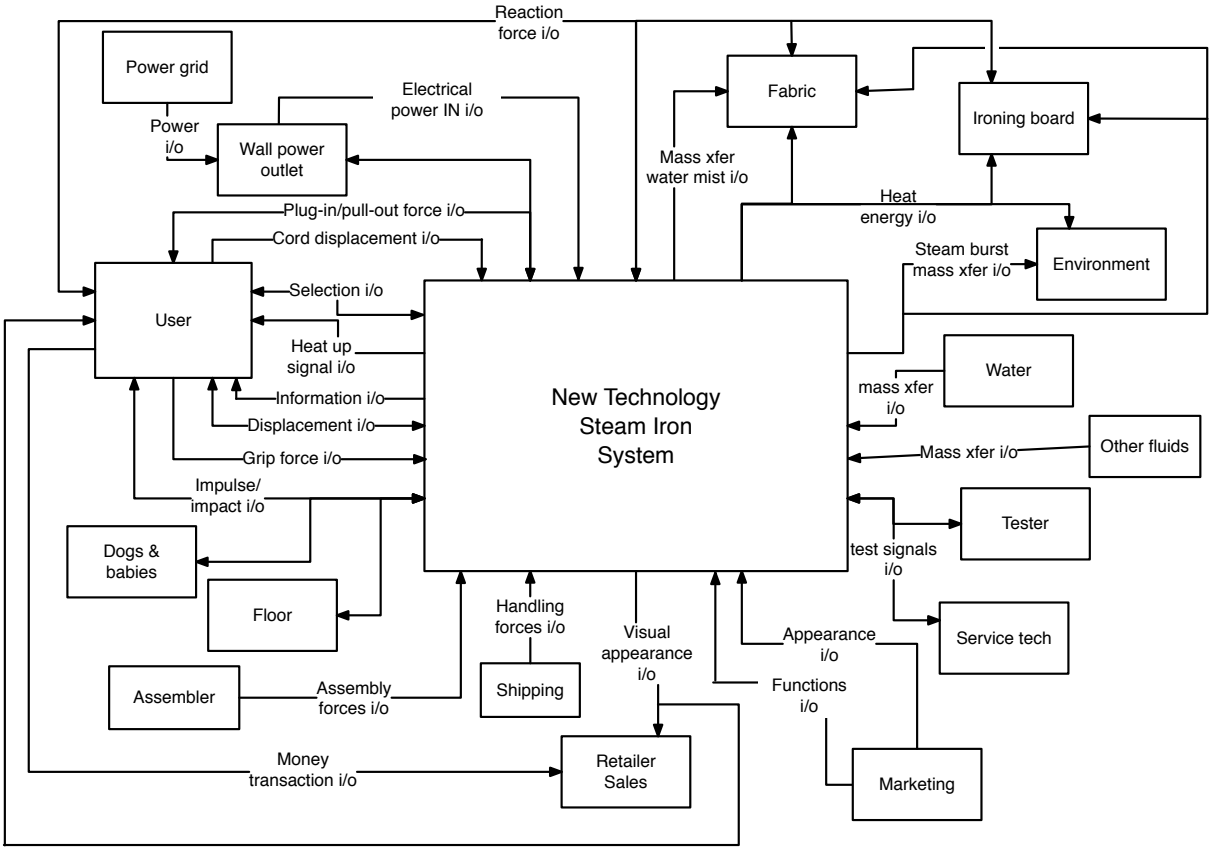


Figure 5 Sample interactions between actors and a digitally controlled steam iron system

Table 5 describes in more detail some of the interactions illustrated in Figure 5. Notice how each interaction is described related back to the actors, pertinent I/Os and requirements. One might discover in the process of documenting the product or process's interactions that actors, I/Os or requirements are missing. It is crucial to revisit earlier system view to add the missing elements.

Table 5 Sample descriptions of interactions for a digitally controlled steam iron

Interactions	Description	Actors	I/Os	Features/ requirements
Plug in/pull out power plug	The user pushes/pulls the power cord into/from the wall power outlet	User Wall power outlet	Plug-in/unplug forces Electrical power IN	Easy to use Standard VAC

Interactions	Description	Actors	I/Os	Features/ requirements
Select fabric settings	The user selects the appropriate temperature for target fabric	User	Selection Information	Easy to use Modern UI Cool look & feel Heat up signal Accurate temperature
Select burst of steam	The user selects the burst of steam mode while de-wrinkling fabrics	User Water Environment Fabric Ironing board	Selection Steam burst mass transfer	Easy to use Modern UI Cool look & feel Burst of steam
Select continuous steam mode	The user selects the continuous steam mode for horizontal and vertical de-wrinkling	User Water Environment Fabric Ironing board	Selection Steam burst mass transfer	Easy to use Modern UI Cool look & feel Vertical steam Heat up signal
De-wrinkle fabrics	The user maneuvers the iron horizontally across fabrics supported on an ironing board with cover to press and steam out wrinkles	User Water Environment Fabric Ironing board	Grip force Heat energy Steam burst mass transfer Displacement	Reliable Safe to use Easy to use Modern UI Cool look & feel No drip Heat fast Cool fast Heat up signal Accurate temperature

Functional decomposition

After the interactions are defined, the next step is to follow each system I/O to a functional element inside the boundaries of the system and begin the process of decomposing the high level functions into smaller functional blocks. Table 6 shows a few sample functions for the digital steam iron system. Note that the functions are intentionally decoupled from potential implementations to provide the maximum flexibility in the search for solutions. These functions can be further refined into smaller sub functions, with internal I/Os developed to connect the flow of materials, energy and signals.

Table 6 Partial documentation of functional blocks within the digital steam iron system

Function	Description	I/Os	Feature(s)
Combine pieces mechanically & distribute forces	The enclosure and housing for the iron system must withstand *impulse* loads when dropped and protect internal hardware from damage during use and shipping by minimizing *forces* to *all* internal components. Must require minimal *assembly forces* during manufacture and testing.	Impulse Assembly forces Grip force Reaction force	Durable Reliable Easy to assemble
Contain water	Provide a means to load and store water for producing *steam burst, continuous steam, fabric misting*.	Water mass transfer	Burst of steam Vertical steam Low effort mist

Function	Description	I/Os	Feature(s)
Control water	Provide a means to interpret signals and direct water to *steam* producing subsystem, *prevent drips and leaks* from the soleplate orifices, and increase pressure for *fabric misting*	Water mass transfer	Burst of steam Vertical steam Low effort mist
Convert electrical energy into heat energy	Provide a means to convert the *alternating current (electrical power IN)* provided at the *wall power outlet* into *heat energy* for the soleplate and the *steam* producing subsystem	Electrical power IN Heat energy Steam burst mass transfer	Heat fast Accurate temperature

Figure 6 shows a sample functional decomposition for the *convert electrical energy into heat energy* function. New I/Os, internal to the system are now defined. Previously, the only I/Os defined were between the actors and the boundaries of the system model. Each function within the system should be documented in this way and combined into a complete functional architecture representation. This functional architecture will later be refined into physical and/or management architectures (collectively referred to as a product architecture).

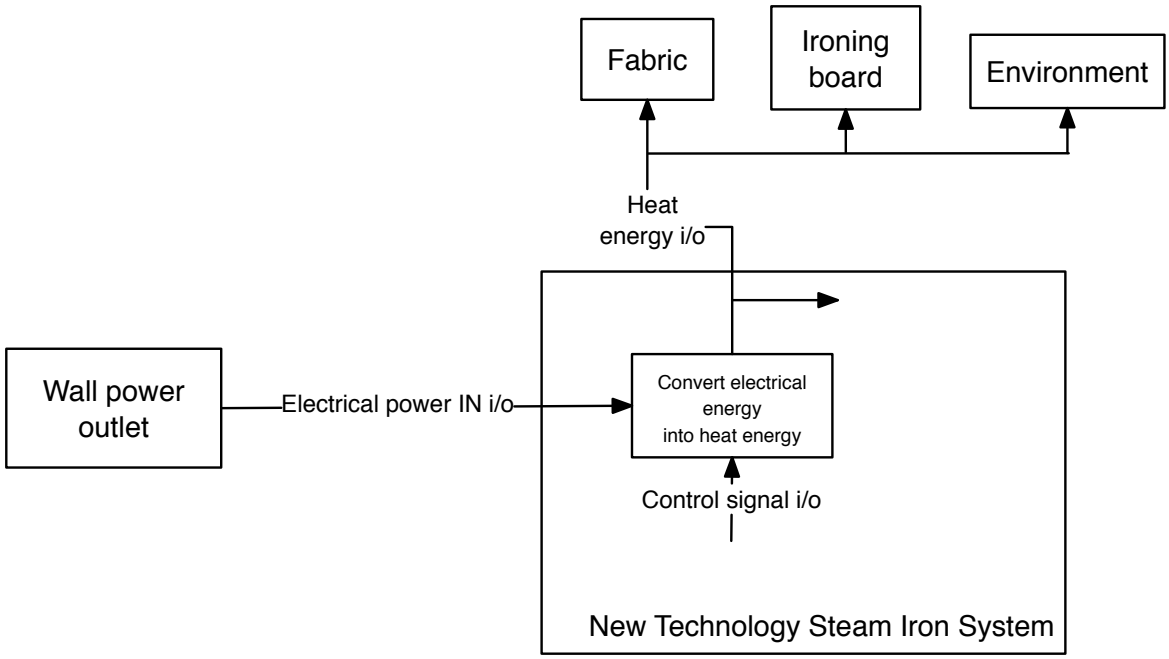


Figure 6 Functional diagram for the *convert electrical energy into heat energy* function

Figure 7 shows the complete functional architecture of the steam iron system. This diagram is the result of consolidating all the functional diagrams with the interactions model.

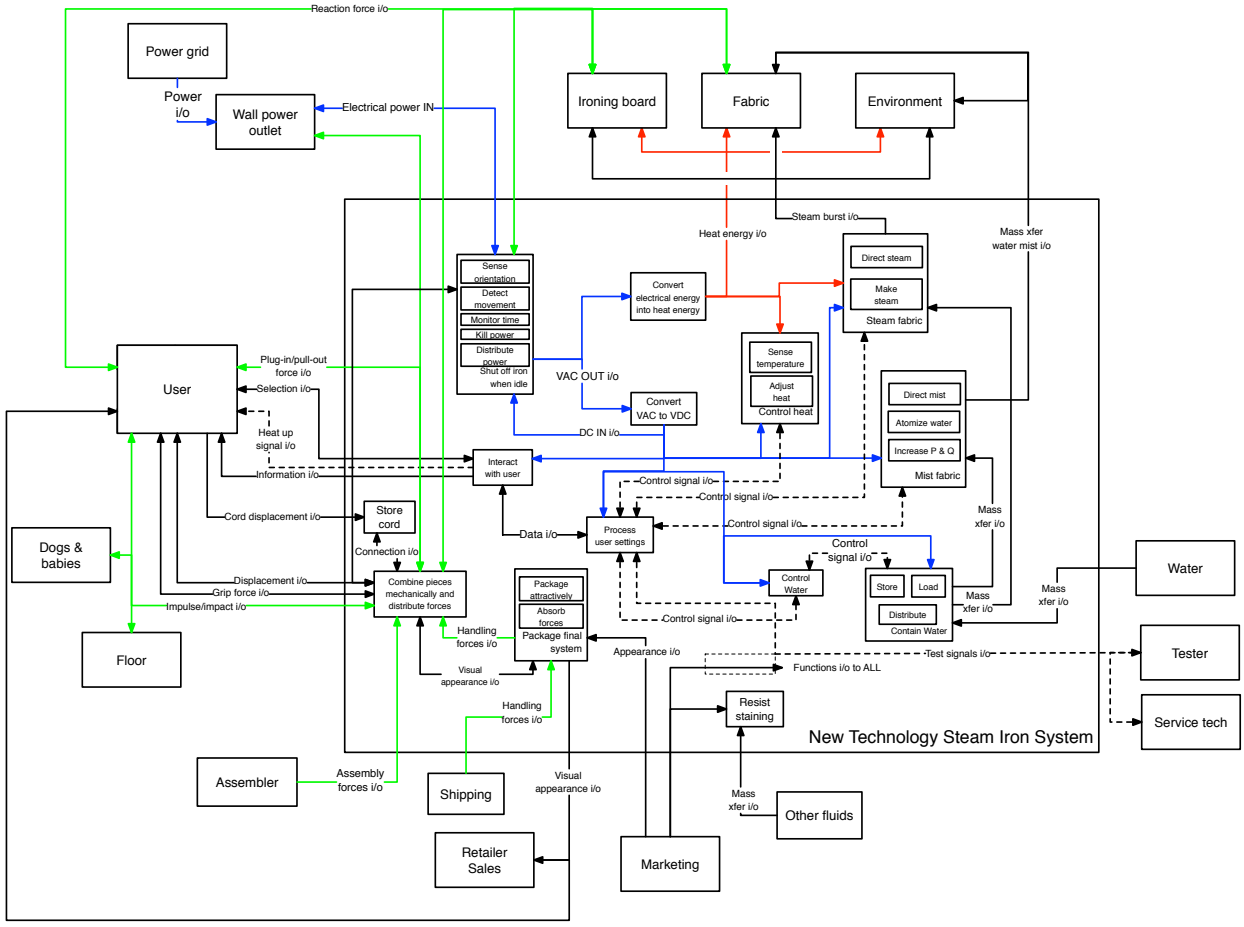


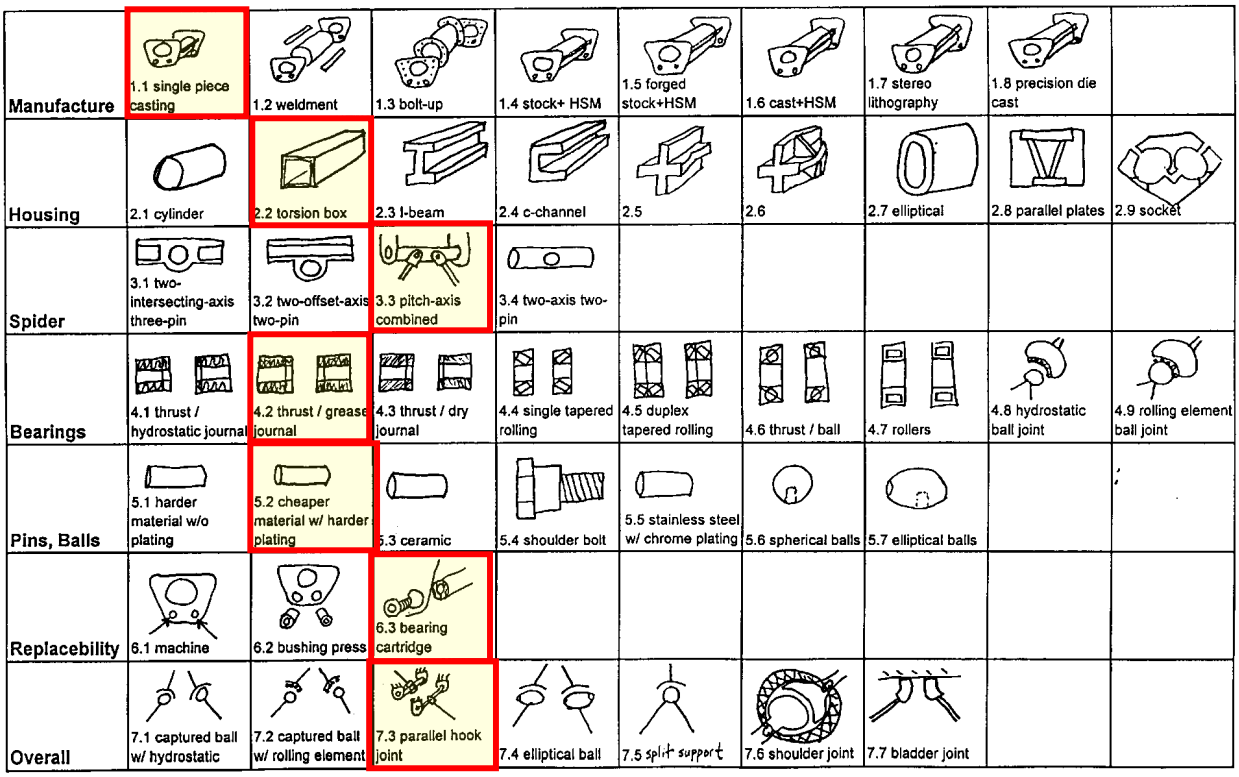
Figure 7 Complete functional architecture model for the digital steam iron

After defining the functional architecture, the team must identify from three to seven critical functions to use as the basis for concept generation. For the steam iron, the following functions were identified:

- Control heat
- Control water
- Interact with user
- Process user settings

With critical functions now identified, the team can commence with internal and external searches to identify multiple solutions for each of these functions. Each of the functional solutions should be numbered and captured in sketches, pictures or other graphical or textual representations. Once defined, the solutions (also referred to as solution fragments or partial designs) can be arranged into a concept combination table to generate system concepts. Figure 8 shows a sample concept combination table for a six degree of freedom parallel platform knuckle joint from a full-scale flight simulator. One concept is shown as the combination of one solution fragment from each of the critical functions (also referred to a critical sub problem) categories.

Note that each solution fragment is numbered so that a concept can be referred to as a combination of the numbers, i.e. 1.1+2.2+3.3+4.2+5.2+6.3+7.3 defines one concept.



One concept = 1.1 + 2.2 + 3.3 + 4.2 + 5.2 + 6.3 + 7.3

Figure 8 Concept combination table for a 6 DOF parallel platform knuckle joint

The concept combination table is just one way of organizing solution fragments. A concept classification table can also be utilized. The key point is to be systematic in the organization of the concepts.

Once organized, concepts for the system should be captured in sketches, pictures or other graphical or textual representations.

Concept generation for software

Software

The concept generation process can provide an effective set of tools for software-oriented projects to develop alternatives for elements such as user experiences, algorithmic approaches, and data structures. User experiences might include graphical user interfaces and frameworks. Algorithmic approaches may include optimization routines (Newton's method, steepest decent, ALM, etc.), graphics processing (GPU versus CPU), and search (binary, A*, B-tree, etc.). Data structure alternatives could include considerations such as hash tables, linked lists, and databases (relational and object-oriented).

The functional decomposition approach in concept generation provides a head start on development of the preliminary **system/product architecture** deliverable.

Presenting this deliverable

This deliverable is presented in two parts, over a two-week span.

Preliminary Concept Generation & Evaluation:

- 1 slide with team name, logo, team members, etc.
- 1 slide with problem decomposition graphic with critical functions (sub problems identified)
- 1 slide with evidence of external search (such as patents, competitors, web, lead users, experts) and solution fragments for critical functions (sub problems)
- 1 slide with evidence of internal search (individual & group sessions) and solution fragments for critical functions (sub problems)
- 1 slide with systematic exploration--concept combination table, concept classification table, etc. illustrating overall system solutions

Concept selection

Hardware **Software** **Process**

The concept selection process is an important part of concept generation. During concept selection, concepts will be compared to each other using decision matrices. In the process of comparing alternatives, new patterns may become evident that will lead to new and better concepts (concept selection is therefore considered a *generative* process). The ultimate goal of this phase is systematically eliminate the least promising concepts until only a few concepts remain. These final few concepts will likely be built (prototyped) and tested so that a final concept can developed into a product for the customer.

Concept screening

Concept screening matrices are used to reduce a large number of concepts down to a manageable number. Selection criteria are used to evaluate each design. These criteria are directly related to the customer needs or product/process design specifications. Once the selection criteria are agreed upon, a decision matrix is prepared. This approach can be applied to concepts for a particular function or for the entire system.

Figure 9 shows a concept screening matrix for the control water function within the digital steam iron example. The first column includes the selection criteria, while the subsequent columns represent system concepts A through E. A reference concept must be established as a benchmark for comparison. The example shows concept B as the reference. Note that the rating for all of concept B's selection criteria are set to zero (0). This concept was considered to be the average concept. For each criterion, the design concept is compared to concept B. If the concept is superior for a particular criterion, then it rates a "+," while inferior concepts rate a "-." Concepts that have similar performance characteristics are rated as "0" or if preferred, "S" for same. The plusses, minuses and zeroes are summed up to determine which of the concepts will be carried

forward for further development or selection. In this example, concept E is dropped from consideration due to its poor performance.

Control Water	A	B	C	D	E
Selection Criteria	1 motor/pump 2 on/off valves continuous	1 motor/pump 2-way valve discrete	1 motor/pump 3-way valve discrete	2 motor/pump direct connect discrete	2 motor/pump 2 on/off valves continuous
Cost	+	0	0	0	-
Reliability	-	0	0	0	-
Noise	-	0	0	0	-
Compact size	0	0	0	0	-
Flow control	+	0	+	+	+
Power consumption	0	0	0	-	-
Net +/-	**0**	**0**	**1**	**0**	**-4**
Total +	2	0	1	1	1
Total -	2	0	0	1	5
Total 0	2	6	5	2	0
Rank	**2**	**2**	**1**	**2**	**3**
Continue?	**Yes**	**Yes**	**Yes**	**Yes**	**No**

Figure 9 Concept screening matrix for the control water function on the digital steam iron system. Concept B is the reference concept.

Before moving forward to a scoring matrix, the team should see if any new concepts can be created from combining the best elements of the designs. Once satisfied, the remaining concepts can be rated with numbers to select the most promising concept(s) for further development.

Concept scoring

Concept scoring is used after the least promising concepts are pruned from consideration. The scoring matrix is similar to the screening matrix in that the concepts are compared to references for each selection criterion. The key differences are that selection criteria are weighted, the reference concepts are selected by criterion, and a numerical rating (1 to 5) is selected for each concept-criterion pair. Figure 10 shows a concept scoring matrix for the control water function. The weightings for the selection criteria were developed from the priorities established by the customer. The reference concepts for each criterion are as follows: cost (A), reliability (B), noise (B), compact size (B), flow control (A), and power consumption (B). Based upon the scoring, concepts A and B were eliminated.

		A		B		C		D	
Control Water Selection Criteria	Weight	1 motor/pump, 2 on/off valves, continuous		1 motor/pump, 2-way valve, discrete		1 motor/pump, 3-way valve, discrete		2 motor/pump, directly connected discrete	
		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Cost	30%	3	0.9	3	0.9	3	0.9	3	0.9
Reliability	20%	2	0.4	3	0.6	3	0.6	3	0.6
Noise	10%	1	0.1	3	0.3	3	0.3	2	0.2
Compact size	15%	3	0.45	3	0.45	3	0.45	2	0.3
Flow control	20%	3	0.6	3	0.6	4	0.8	5	1
Power consumption	5%	2	0.1	3	0.15	3	0.15	2	0.1
	Total Score		2.55		3		3.2		3.1
	Rank	**4**		**3**		**1**		**2**	
	Continue?	**No**		**No**		**Yes**		**Yes**	

Figure 10 Concept scoring for concepts to address the control water function

The comparisons need to be based upon data—whether this data is gathered from published specifications or from testing. Sometimes final concept selections cannot be made until after prototypes are built and validated by the project sponsor. Prior to completing the concept selection phase, the team should closely review the scoring matrix and see if new promising concepts can be created from the best elements of other concepts.

Preliminary Concept Selection:

- 1 slide with team name, logo, team members, etc.
- 1 slide with problem decomposition graphic with critical subproblems identified
- 1 slide with systematic exploration—concept combination table, concept classification table, etc. illustrating overall system solutions
- 1 slide with concept screening results—including new concepts generated and winners
- 1 slide with concept scoring results
- 1 slide with recommended concepts for further development (one to three)

System/product architecture

Software

Overview

The objective of this deliverable is to provide a clear overall system/ product organization including a good architectural overview and justification. The total system/product is broken down into well-defined modules including their functionality and interfaces to each other. All functions listed in the requirements must be covered neither by too many nor too few modules.

The architecture should be designed to accommodate likely changes within a module without affecting the rest of the modules. Major data structures need to be described and justified. Database organization and content must be specified.

A clear architecture with well-defined modules allows delegation of specific modules to development engineers for parallel or concurrent development of a product. The functional architecture developed in the concept generation deliverable will minimally satisfy the requirements of this deliverable.

System/product architecture checklist:

1. Is the overall program organization clear, including a good architectural overview and justification?
2. Are the components (modules and objectives) well defined including their functionality and interfaces to other components?
3. Are all the functions that are listed in the requirements covered?
4. Is the architecture designed to accommodate likely changes?
5. Are all major data structures described and justified?
6. Is the database organization and content specified?
7. Have all the dependencies been identified?
8. Is the user interface modularized so that changes in it won't affect the rest of the program?
9. Are key aspects of the user interface defined?

Software development should only proceed after satisfactory answers to the questions in this checklist have been provided or actions to resolve outstanding issues have been defined with a date for resolution.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide with a graphic showing where system/product fits within existing environment--identifying all interfaces
- 1 slide with system/product broken down into well-defined modules with interfaces between modules (UML or ERD) and functions each module supports
- 1 slide describing major data structures
- 1 slide with sample user interface

Configuration management plan

Software

Configuration management is a process of identifying the configuration (parts) of a product for the purpose of systematically controlling changes to the part.

The configuration management plan defines the goals, structure, responsibilities and degree of change control. The plan is created during Phase I of the IPPD Software Development process and defines the following:

- the baselines to be established during the development cycle
- the objectives to be included in each baseline
- database, library structure, responsibilities and tools to be used for managing changes
- a set of change control criteria for
    - level / method of authorization
    - type of verification
- change control criteria such as
    - changes of the design or specifications
    - cause of change (errors, enhancements, requirement changes)
    - impact on lines of code, documentation, work items, personal hours

Baselines include those work items that have been verified and agreed upon to serve as the basis for further development. Once the basis is established no work item may be added or modified without appropriate authorization.

Teams using Scrum should adapt this deliverable to support their process. For example, the team should focus on elements such as naming conventions for source code, how the Git repository is organized, the level of detail for Git commits, and how software versions will be identified.

Presenting this deliverable:

- 1 slide with team name, logo, team members, etc.
- 1 slide identifying baselines for software builds & objectives for each build
- 1 slide with change control criteria (numbering & naming conventions)
- 1 slide enumerating what objects under change control
- (i.e. requirements, specs, architecture, code, test plan, etc.)

Project plan

Hardware **Software** **Process**

Overview

The project plan is required to plan and organize the design effort through the fall and spring semesters. The Project teams must estimate the physical and financial resources needed to complete the project and provide a method of tracking progress towards completion. Due dates for project deliverables and reviews with industry sponsors will be part of the project schedule. The project plan is dynamic and must be continuously revised and tracked with current progress. The initial plan and all subsequent revisions will be reviewed with the industry sponsor. Use the project roadmap as a starting point for the task sequencing and dependencies.

How to create a plan:

1. Set a clear endpoint for the project and work backward.
2. Determine due dates for major deliverables.
3. Determine activities required for deliverables and the relationship among activities.
4. Estimate time required for each activity.
5. Create a picture of the schedule. Use Microsoft® Project to develop your plan.
6. Plan by week.

7. Overlay your sponsor's development process and deliverables or checkpoints onto your plan.
8. Modify the plan as you improve your understanding of the project.

Important considerations: A good project plan will include more than just a schedule of the deliverables. The plan should include the following:

- training for design systems and software tools (CREO, SolidWorks, Mentor Graphics, Altium Designer, Visual Studio, Minitab)
- timeline for fabrication
- timeline for testing
- timeline for troubleshooting (debugging)

Project risks

A project plan is not complete without an assessment of the project risks. Risks broadly fall under the following categories:

- technical
- schedule
- manufacturing
- procurement
- financial
- legal

Risks are assessed by rating the negative impact potential (high or low) on the project if the risk occurs and the probability/likelihood of the risk occurring (high or low). A simple way to rate risks is to use the following rule:

Risk Rating	Impact	Probability	Risk Code
Very High Risk	High	High	HIHP
High Risk	High	Low	HILP
Medium Risk	Low	High	LIHP
Low Risk	Low	Low	LILP

Project success is highly dependent upon the team's ability to mitigate risks. Risk mitigation requires specific actions that will move the risk rating for a particular project element from a higher risk rating to a lower risk rating, with the goal of reducing all major project risks to a low risk rating. A series of mitigation steps may be required to move a high risk to a low risk. Examples of mitigation strategies include the following:

- devising a test to prove that a component or subsystem functions properly
- validating the work flow of a software tool with a user interface demonstration
- procuring back-up parts in case of failure during testing
- performing a system simulation prior to fabrication and physical testing
- developing a series of prototypes for testing and customer validation prior to final design/coding/fabrication

A list of ongoing project risks should be captured and tracked. A convenient way to represent these risks is to create a risk table, also known as a risk register. A sample risk register is illustrated below.

No.	Risk	Rating	Code	Mitigation Strategy	Owner
1	Shaft seal will leak	Medium	LIHP	verify shaft diameter with micrometer prior to ordering seal	Jane
2	Soon-to-be-released daughter board drivers will not be available on schedule	Very High	HIHP	1. work closely with the supplier to obtain beta releases; 2. identify an alternative daughter board	Barry

Plan example

An example project plan based on current IPPD project deliverables is illustrated in the "Creating a Project Plan" document, which can be found in Canvas in the "Resources" section, in the "How To" folder. Potential areas of project plan execution risk are defined as well.

Important: The project plan is not a "do-it-once-and-forget-it" deliverable. It is expected that the Project team will consult and update the plan on regular basis (weekly). The plan is a tool to help manage the project. "Manage the tool; don't let the tool manage you."

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 to n slides with project Gantt chart with major project milestones and deadlines
- 1 slide with a sample breakdown of 2 deliverables--show dependencies between tasks, show team & sponsor tasks--highlight tasks that are project-specific (not a generic IPPD deliverable)
- 1 slide with a risk table showing description and severity, ordered by severity

Business case

Hardware Software Process

Overview

All design projects must include a business case analysis; however, not all projects are undertaken purely for economic reasons. Some projects are undertaken just to maintain parity with a competitor or to comply with new regulatory requirements. Department of Defense projects in particular may be difficult to assess—sometimes it is sufficient to prove that the chosen design option is the lowest cost among alternatives or show how the design improves reliability or mission effectiveness.

Typically, the business case will include a list of assumptions, a list of the variables (such as increased sales revenue, estimated product cost, estimated development effort, and marketing and support costs), and the calculation of the net present value, internal rate of return, payback period, profit before and after taxes, and return on investment. The internal rate of return should be compared to the sponsor company's hurdle rate.

Sensitivity analysis: It is important to determine how changes in some of your business case assumptions affect the profitability and payback of the project. A business case that is insensitive to variations in the assumptions is said to be robust. The effect of variation in key business case drivers should be explored in a sensitivity analysis. This analysis will provide insight into how changes in these key drivers effect the profitability of your proposed solution. The sensitivity analysis is one tool for managing the uncertainty in the project economic data.

For instance, in a new design or redesign of commercial product, the key business case drivers might be sales volume, unit cost, and gross margin. In a sensitivity analysis, a baseline is first established using best guess values for the key drivers. The net present value, internal rate of return, and payback period are then determined. These calculations are repeated using optimistic and pessimistic values for the key drivers (for instance, sales volume increases or decreases by 50%).

Important: Due to the competition-sensitive nature of certain cost information, such as labor rates and product margins, it may not be possible to create an accurate business case. However, it is still possible and highly desirable to create an accurate business case model. Spreadsheets are particularly well suited to this task. The sponsor can easily replace your estimated values for the key economic drivers with precise figures.

Updates: The business case should be revised and up to date for the System Level and Final Design reports.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide with business case assumptions
- 1 slide summarizing the cash flow statement
- 1 slide summarizing the key business case calculations (including sensitivities)
- 1 slide with recommendations

Prototype plan

Hardware Software Process

Overview

Tom Kelley, General Manager at IDEO says, "a picture is worth a thousand words. Only at IDEO, we've found that a good prototype is worth a thousand pictures." Experience has shown that successful development teams prototype early and often throughout the IPPD process. Each IPPD team is required to develop a technology feasibility prototype for demonstration at the System Level Design Review. The prototype should embody one or more key technical aspects of the system, such as functional behavior of a system/product or module, user interface mock ups, or system breadboards. Major advantages that a prototype can provide include the following:

- a mechanism that allows designers to evaluate alternatives and to investigate problems
- an early assessment of the impact of a new function to reduce risk

- evaluation of changing environmental characteristics
- improved requirements, specifications, design verifications and user interfaces

The prototype plan will be included in the Preliminary Design Report and as such must be approved by both the project coach and sponsor liaison engineer. The plan should be started as early as possible, but no later than the conclusion of the Preliminary Design phase. The plan should a minimum include the following:

- a table indicating what specifications will be demonstrated by the prototype(s)
- a brief description of the prototype(s), the objectives, and important fabrication elements
- a list of resources required and responsible team members
- a schedule that includes acquisition, fabrication, and testing
- a script indicating how the prototype(s) will be demonstrated during the System Level Design Review

Through software prototyping the functional behavior of a system/product or module can be observed early in the development cycle. Rapid prototyping and evolutionary design can be an alternative to the generation and validation of written requirements/specifications as a way of ensuring that a software product will be responsive to user needs. Major advantages that a software prototype can provide include the following:

- a mechanism that allows designers to evaluate alternatives and to investigate problems
- an early assessment of the impact of a new function to reduce risk
- evaluation of changing environmental characteristics
- improved requirements, specifications, design verifications and user interfaces

Software prototype plans should be made early in the development phase and should be determined based on guidelines such as the following:

- What do we need to know about user interfaces (screens, screen transitions, dialogs, reports etc.)?
- What is our plan to evaluate the prototype feedback?
- What do we need to know about data flow, internal interfaces, and feasibility of design or performance issues?

Prototyping results can be used to make a better product. Generally, however, prototype should not be used in the formal product. A prototype by design and intent answers specific questions. Attempting to implement the prototype as the operational product can result in new problems as new questions surface.

Things to avoid when developing a prototype:

- rushing into prototyping before adequately defining technical objectives
- losing sight of prototyping objectives
- developing the prototype into the product
- making endless prototype iterations
- delaying documenting the prototype results

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide with specific questions trying to be answered by one or more prototypes
- 1 slide with list of prototypes, prototype classification (i.e. functional, nonfunctional, aesthetic), tool used for prototyping (i.e. a 4GL, Excel macro, VB application), resources and schedule
- 1 slide with prototype evaluation criteria (how to judge results & what to do with the info)

Preliminary design report and project plan

Hardware Software Process

Overview

(See **Major Deliverables** in IPPD Report Process section.) The preliminary design report is the first major deliverable in the eight-month design project. The report summarizes the project to date and serves as an agreement with the sponsor company that the proposed design solution, project plan, business case will meet the sponsor's needs. Prior to presenting the report during a design review at the sponsor's facility, the report should be thoroughly peer reviewed by project team, coach, and liaison engineer.

A signature page should be included in the report. The Preliminary Design Report is your team's agreement with the sponsor company for the selected design concept, scope of work, and project plan. As such, it is important that the team, faculty coach, and liaison engineer agree upon the content. Therefore, we require a signature approval page be included with the report. **Please see the Signature Page Specification and Examples in Canvas in the Files "Deliverables" folder.**

During the design review, new issues may arise and the project may be rescoped. The report should be revised to reflect these changes and a sign-off from the liaison engineer obtained once all agreed upon modifications are incorporated.

Proposed content and presentation outline

Executive Summary (~1 page)

- Provides a quick overview of the report. The reader should gain a good idea of what the project was all about and what the results were. Key technical specifications, experimental and prototype results, and financial measures should be included. Explicit recommendations should also be stated.

Table of Contents

- Table of Contents
- Reference Documents
- List of Figures
- List of Tables

1.0 Project Assumptions and Objectives

- what the project is, background information
- the scope of the project. What is and what is not to be accomplished
- the major objectives such as
    - reengineering existing product for cost / expense reduction
    - new product family or extension
    - increased market share, etc.
- what the product is expected to do
- annual quantity
- technical strategy (software)
- configuration management plan (software)

2.0 Customer Requirements

- project stakeholders and their wants and needs including the relative importance of each need (or requirement) another
- Product Specifications
- values of measurable engineering characteristics (specifications) of the product as derived from the customer requirements
- relationship of product specifications to customer wants and needs (House of Quality)
- identified Technical Performance Measures (TPM) and current achievement

3.0 Analysis of Competitive Products

- patent and literature search
- benchmarking of competitive products
- customer perception
- technical performance
- costs / price

4.0 Concept Selection and Description

- functional architecture model with description of actors, I/Os, interactions and functions
- preliminary product/system architecture (software)
- demonstration of the comprehensiveness of the concept generation (decomposition external search, internal search, systematic exploration)
- appropriate documentation of concepts (sketches, schematic diagrams, block diagrams, etc.)
- list of evaluation criteria
- documentation of selected concept
    - schematic diagrams
    - block diagrams
    - modeling
    - process layouts
    - evaluation of concept against product specifications

5.0 Prototype Plan

- planned hardware, simulation and software prototypes

- timeline, resources needed and expected costs

6.0 Project Plan, Resources, Schedules

- major checkpoints/deliverables
- list of activities
- time required to complete
- target dates for completion
- responsible team members
- Scrum sprint schedule and burn down charts (software)

7.0 Business Case

- assumptions
- increased sales revenue
- estimated product cost
- estimated development effort
- marketing and support costs
- profit before/after taxes
- net present value
- return on investment

8.0 Issues

- list of areas in the design that are not too well understood and will require additional work
- parts, components, subsystem sourcing for prototypes
- prototype testing

9.0 Recommendations

- list explicitly what you recommend: “approve the selected concept,” “authorize additional prototyping funds,” “provide team access to company systems, facilities, etc.”

Packaging the report: The Preliminary Design Report should be professionally labeled and packaged. It serves as the basis for an oral presentation and design review with the project sponsor company. Prior to presentation at the sponsor site, an 18-minute version of the presentation will be peer reviewed before an audience of other project teams and faculty from the University of Florida.

Important Dates:

Due Date	**Description**
Early to mid Oct.	Peer review (practice session, during class, attendance mandatory)

Mid Oct.	First complete draft of the Preliminary Design Report posted on team wiki site
End Oct.	Preliminary Design Review at sponsor's site
End Oct.	Finalized Preliminary Design Report with signature page uploaded in the *final_documents* folder on the team repository

In the beginning of October, your team should plan a visit to your sponsor to present your Preliminary Design Report. If you have a software-oriented project, you should also include content from your software deliverables. The PDR is your agreement with the sponsor company regarding what you will deliver in April.

The first draft of the Preliminary Design Report is due in mid-October. The report is to be published on your repository. Typically, the sponsor will raise some issues with your report during the design review. It may take several weeks to resolve the issues. Therefore, the final *approved* version of the report is due on end of October. **You can find a PDR example in Canvas in the Files area within the "Deliverables" folder.**

A peer review for the PDR presentation will be held in mid-October. Your team will be assigned a room with three to four other teams. Each team will have 18 minutes to present their PDR and 12 minutes for Q&A. This session will be a dry run of a more comprehensive presentation to be delivered live to your sponsor. Be sure to have your liaison approve the content you plan to share on the prior to Peer Review. You may not divulge proprietary information during this presentation.

Note: *the peer review presentation is a subset of the presentation that you will present to your sponsor.*

Preliminary Design Report Peer Review

The Preliminary Design Report needs an 18-minute PowerPoint presentation. A **Peer Review** is an 18-minute PowerPoint presentation summary of your report, followed by feedback from your peers and faculty coaches. Peer Reviews can be for the Preliminary Design Report, System Level Design Report, and Final Design Report. This presentation is intended to be a subset of the content to be presented by the team at the sponsor's facility.

Product architecture

Hardware **Process**

Overview

The objective of this deliverable is to transform the functional architecture elements into a physical architecture and then each functional element in terms of parts, components and/or subassemblies. The process of developing the product architecture will serve to partition the product (or process) into physical elements (blocks) and define the interfaces between each element.

The process of decomposing the functional requirements of the product during the concept generation phase results in a functional representation of the product system—the functional architecture. This functional architecture does not consider how the functions might be grouped, implemented in physical elements or grouped geometrically. The transformation of the functional architecture into a physical architecture involves arranging the functional elements into modules/units (also referred to as "chunks" or "blocks") created to implement the functions. The distribution of the various chunks determines the product architecture.

Coupled with each chunk there are associated interfaces that may be required for connections to other modules. In addition, each chunk may have a technical risk associated with the development. In developing the architecture, the team must therefore address issues such as concentration/location of chunks, interfaces and risks. Each of these decisions can have implications for the product development speed.

The product architecture is developed through the following steps:

1. create a schematic of the product's physical elements
2. cluster the schematic's physical elements into chunks
3. create a rough 2D or 3D geometric layout of the chunks
4. identify the fundamental and incidental interactions between the chunks

Create the schematic

Starting with the functional architecture diagram, develop a schematic of the physical elements that will implement the product's functions. Figure 11 shows a schematic of the digital steam iron that implements the functional architecture depicted in Figure 7. Notice how the schematic reflects design decisions made during the concept generation and selection process. The function of control heat, with its sub functions of sense temperature and adjust heat, are now implemented with a thermistor to sense the temperature, and a relay controlled by a microcontroller to adjust heat. Detailed specifications for each of these physical elements will be determined once the physical (product) architecture is completed.

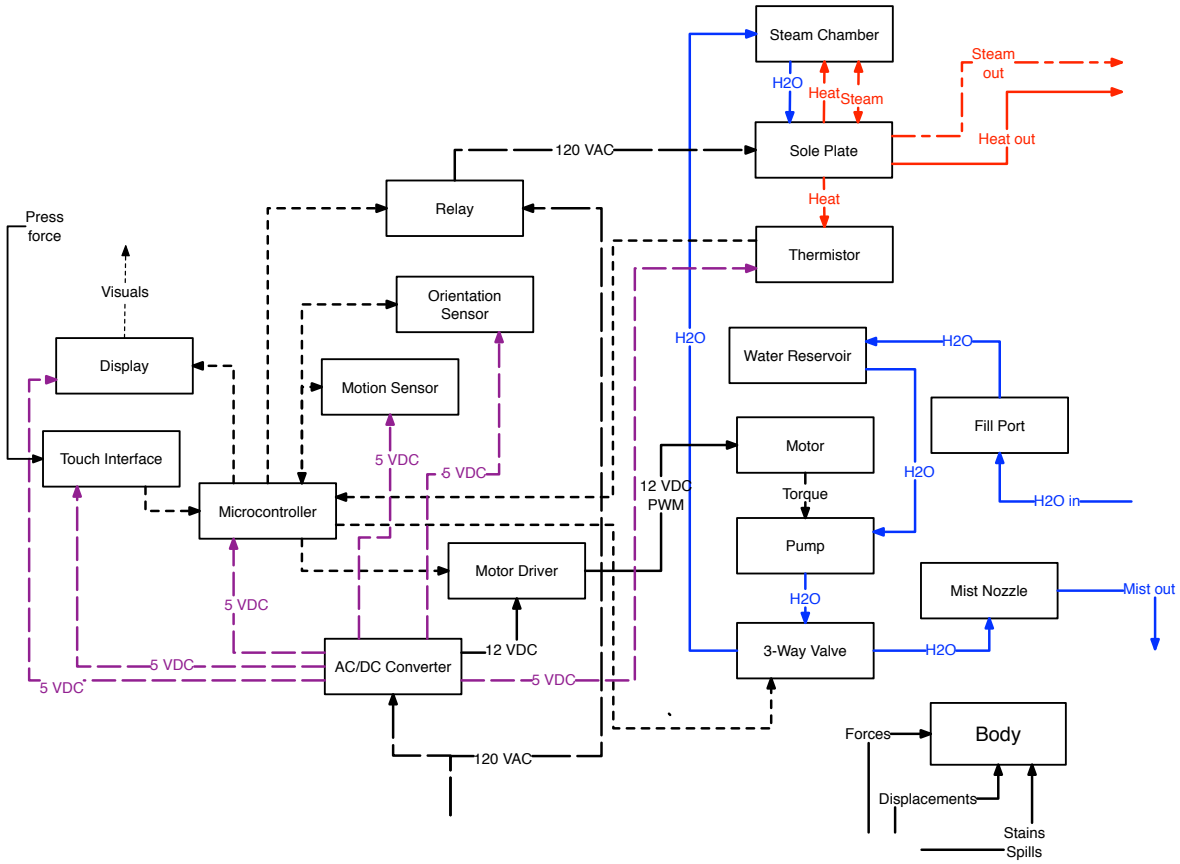


Figure 11 Schematic of the physical elements that implement digital steam iron functional architecture

Table 7 shows a partial list of the components and subsystems of the physical digital steam iron schematic. The table is necessary to ensure that all functional elements have been captured by physical components or sub systems.

Table 7 Subset of components and subsystems mapping into digital steam iron system functions

Component/ sub system	Description	Physical I/Os	Functions
Display	Display iron status and user selections for fabric temperature, continuous and burst of steam, and low effort mist	5 VDC in Data in Visuals	Interact with user
Touch interface	Provide a means to turn the iron on/off and to input user selections for fabric temperature, continuous and burst of steam, and low effort mist	5 VDC in Data out Press force	Interact with user

Component/ sub system	Description	Physical I/Os	Functions
Microcontroller	Process user selections, display iron settings and status, control heat and water, provide PWM, and manage auto shut-off	5 VDC in Data out Data in	Process user settings Monitor time Control heat Adjust heat Control water Kill power
AC/DC converter	Convert 120 VAC wall power in to 5 and 12 VDC outputs	120 VAC in 12 VDC, 1A out 5 VDC, 30mA out	Convert VAC to VDC
Motor driver	Convert PWM data stream from microcontroller and 12 VDC, 1A input to high power PWM output to operate the pump motor	Data in 12 VDC, 1A in 12 VDC, 1A, PWM out	Control water Increase P & Q

Cluster the elements

The physical elements are now assigned to chunks (or blocks). In a modular architecture, each physical element may be assigned to an individual chunk. In an integral architecture, multiple elements are assigned to each chunk. Figure 12 shows the PCB and auto shut-off chunks side by side with the schematic representation. Notice how the auto shut-off chunk is completely contained within the PCB chunk.

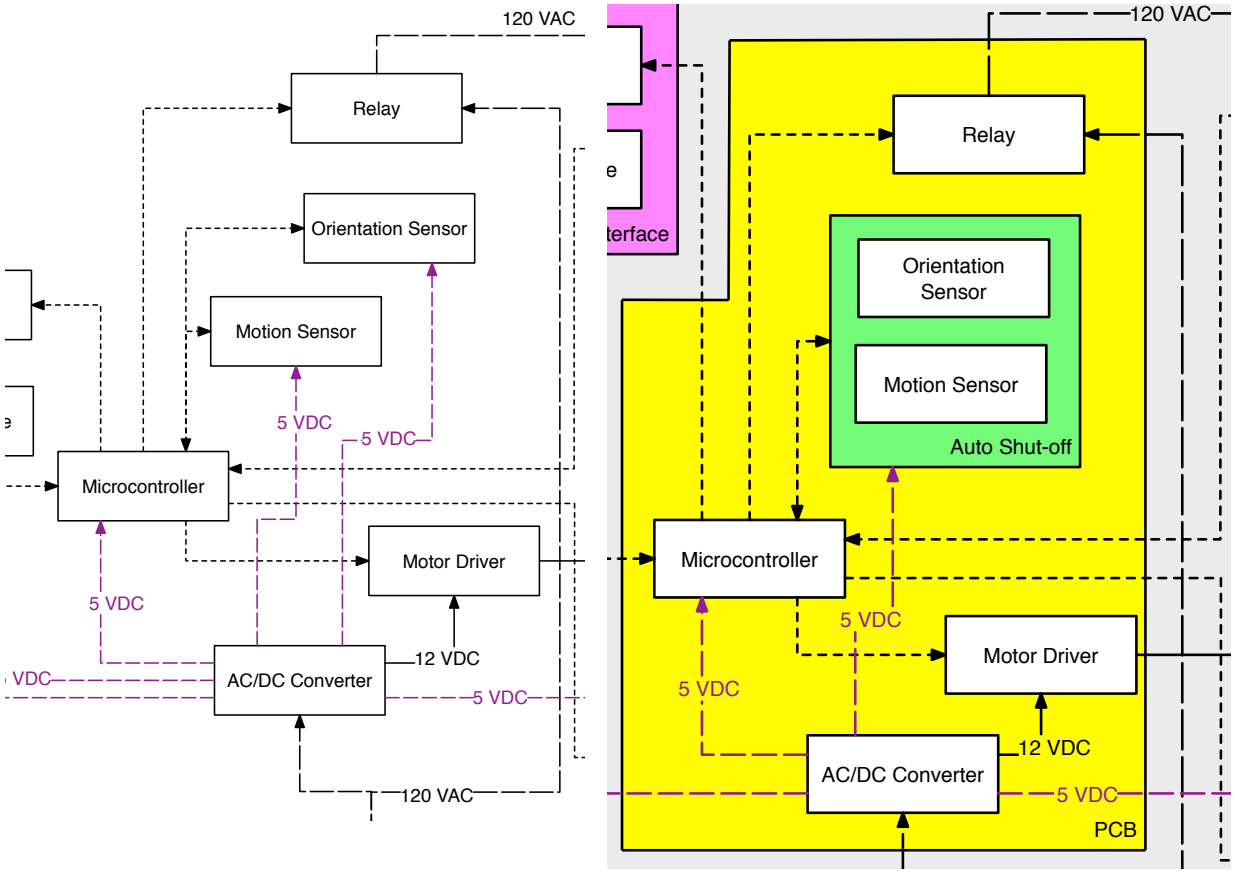


Figure 12 The PCB chunk, on the right, is formed from multiple physical elements. The auto shut-off chunk is contained within the PCB chunk.

Figure 13 shows the chunks developed for the digital steam iron. The body chunk holds all the other chunks. Some elements, such as the water reservoir, fill port and mist nozzle, are integrated within the body rather than implemented as part of other chunks. This is an example of a design decision made by the development team.

It is possible that certain functional elements will have to be shared across numerous chunks. For example, a safety system may be deployed across multiple blocks.

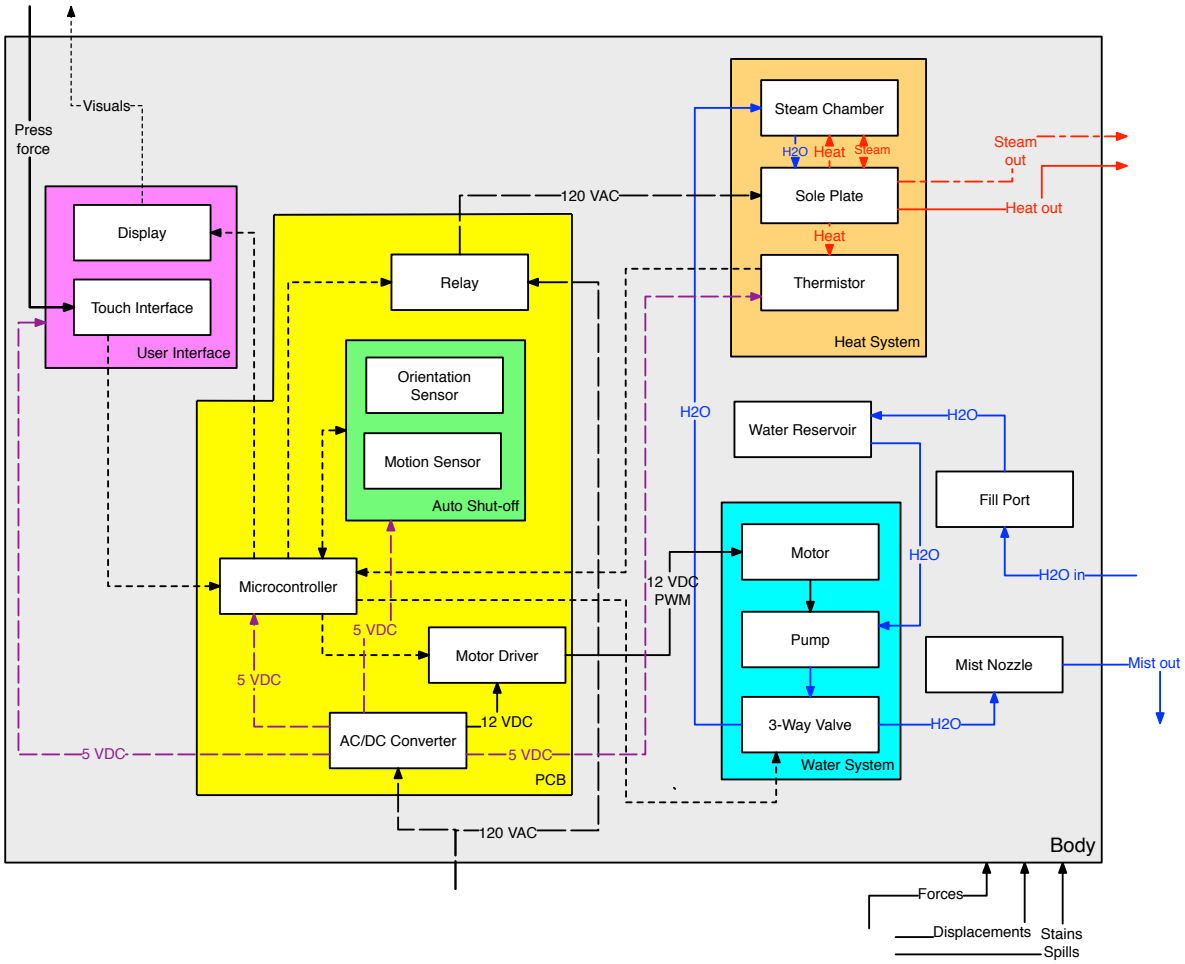


Figure 13 Entire physical architecture of the digital steam iron

Table 8 shows how the physical elements of the digital steam iron are implemented in physical chunks/block. The table shows a partial implementation of the mapping. The table is necessary to ensure that all physical elements have been captured by one or more chunks.

Table 8 Partial mapping of digital steam iron components/sub systems into chunks

Chunk/block	Description	Physical I/Os	Components/ sub systems
Auto shut-off	System senses the orientation and movement of the body	5 VDC, 30 mA in Data out Displacement	Orientation sensor Motion sensor
PCB	Contains primary electronics of the system, including the microcontroller, power management, and water and heat control	5 VDC, 30mA out 12 VDC, 1A PWM out 120 VAC, 15A in 120VAC, 15A out Data in Data out Data in (analog)	Microcontroller AC/DC converter Motor driver Auto shut-off Relay

Chunk/block	Description	Physical I/Os	Components/ sub systems
User interface	Includes a display for iron status and user settings, and a touch interface for capturing user inputs such as power on, temperature setting, steam setting, and misting	5 VDC, 30mA in Data in Data out Press Visuals	Display Touch interface

Create a rough layout

Once the chunks are defined, the next step in creating the physical (product) architecture is to create a rough geometric (2D or 3D) layout of the system. Some chunks may need to be adjacent to each other to minimize signal or other losses at the interfaces. For example, pressure drops may occur if fluids have to travel through many connections over long distances, or using long structures to transmit mechanical signals or loads may lead to loss in fidelity due to excess deflections or damping. In many cases, circuit board locations may be extremely flexible due to the ability to transmit voltage and current over wires with few losses.

Figure 14 shows a rough 3D layout of the digital steam iron concept. The layout provides a notional arrangement of the iron's physical elements. This layout can be used as a focal point for discussions with product stakeholders to communicate product intent and gather feedback.

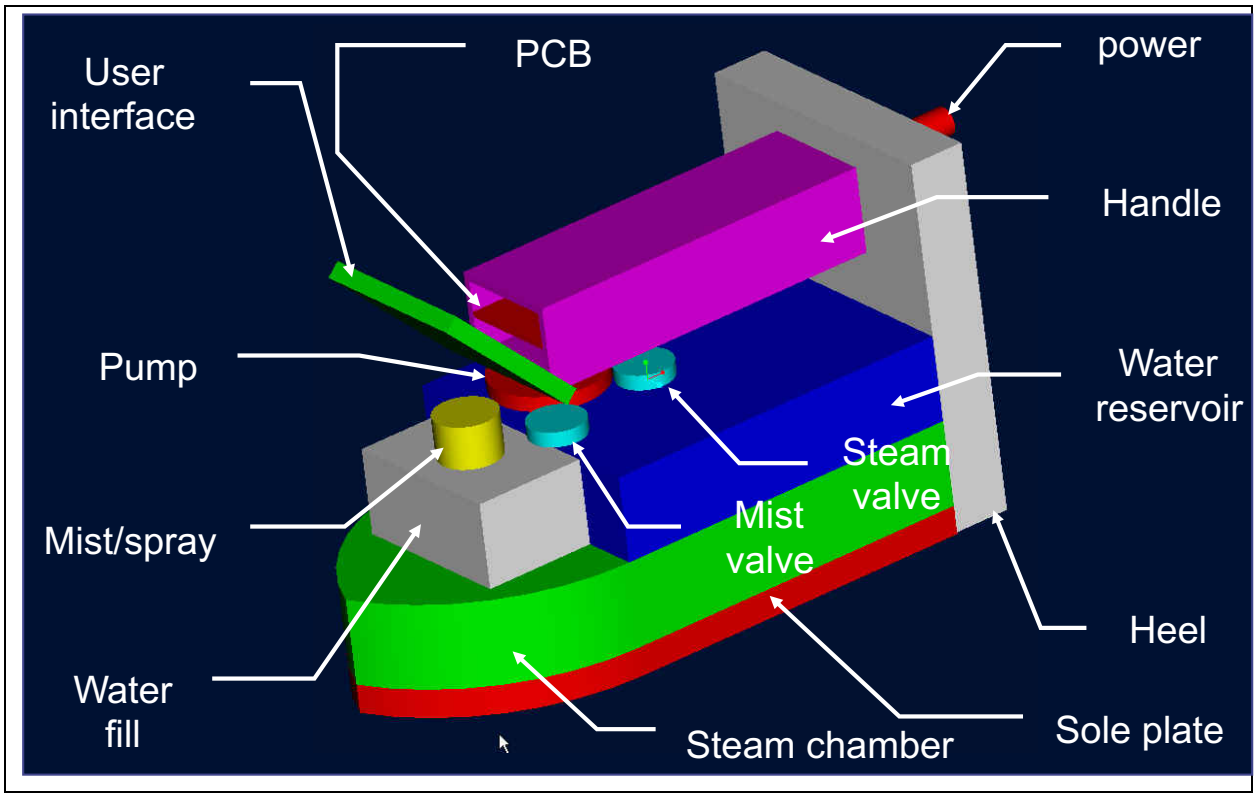


Figure 14 A rough geometric layout of a digital steam iron concept

Identify the interactions

As part creating the physical architecture, the interactions between the chunks are provided in a diagram depicting the intended and incidental interactions. Intended interactions are the desired interactions that occur at the interfaces between the chunks. The incidental interactions are undesirable interactions that may occur at the interfaces between the chunks. In the steam iron example, heat energy from the soleplate to the fabric is a fundamental interaction, while heat energy to the process user settings chunk would be an incidental interaction. Other examples of incidental interactions include noise, electromagnetic interference, radiation, water leaks and vibrations. Understanding the fundamental and incidental interactions allows the design team to refine physical arrangements (i.e. move the noise-sensitive elements away from the noisy elements) and countermeasures (such as shielding, damping or insulation).

Figure 15 shows the incidental interactions that may occur between the digital steam iron chunks. Heat, steam and leaking water are major concerns. Clearly, electronics need to be physically separated from the sources of these incidental interactions. If physical separation is not possible, then the design team may consider potting the electronics in a water resistant and heat tolerant polymer.

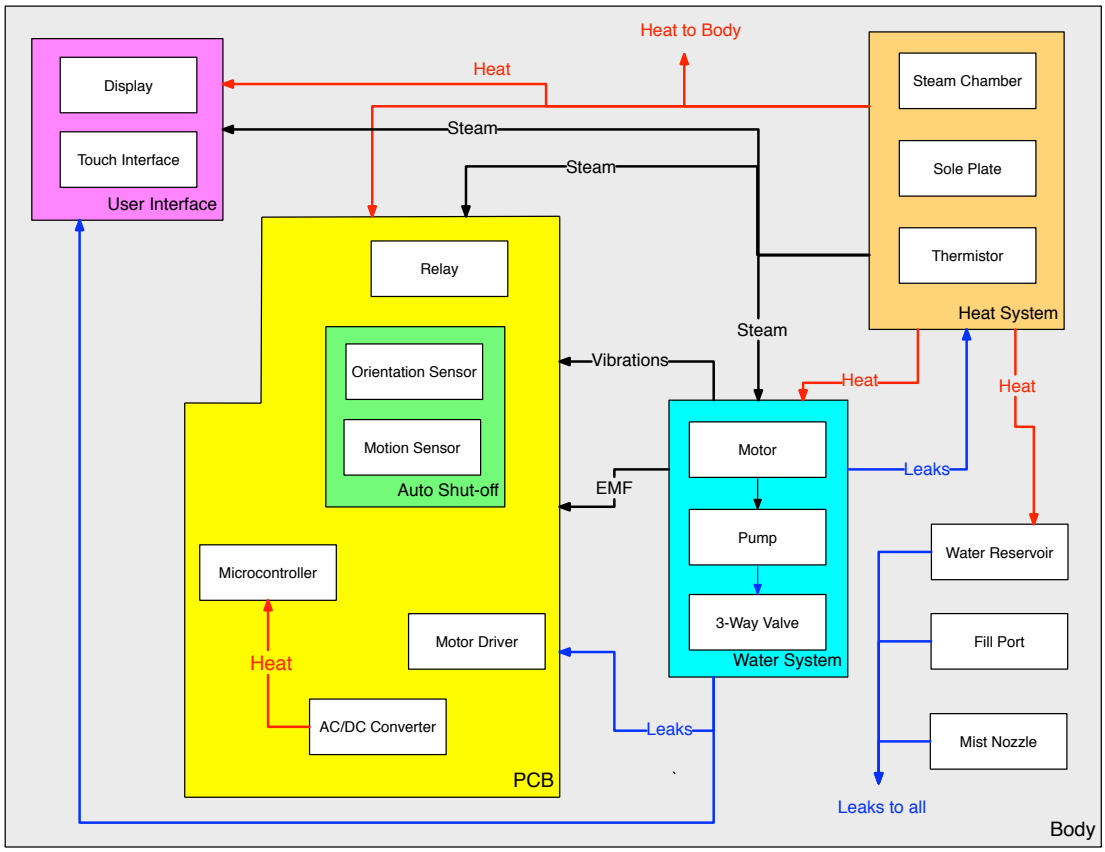


Figure 15 Incidental interactions between digital steam iron chunks

Table 9 shows a partial list of incidental interactions between various digital steam iron chunks. The mitigation strategies provide guidance in minimizing the impact of the incidental interactions on the performance of the system. The design team should identify and prioritize mitigation strategies. Testing may be required to overcome the incidental interactions. This preliminary work on incidental interactions may provide insights for future studies on Failure Modes and Effects Analyses (FMEA).

Table 9 Partial list of incidental interactions between digital steam iron chunks

Incidental interaction	Description	Chunks	Mitigation strategy
Heat	Voltage converter heat may damage the microcontroller or other circuits on the main logic board	AC/DC converter Microcontroller	Isolate the AC/DC converter onto a separate power board that can be located away from the logic board. The power board might also contain the relay for the control heat function.
Heat	Heat from the sole plate or steam chamber may damage/warp the body or other components such as the reservoir, user interface or PCB	Heat system Body Water reservoir Water system Display PCB	• Select high temperature plastics for the body • thermally insulate the hot parts from the cool parts
Leaks	Leaks from the water system, reservoir, fill port or mist nozzle may damage the circuits or user interface components	Water system Reservoir Fill port Mist nozzle PCB User interface	• Keep water-sensitive components isolated from the water source • select water-resistant components • gasket and seal areas around water-sensitive components • Pot the PCB in epoxy

Modularity leads to rapid progress

Modularity, if done well, allows shortening the development cycle, because modules can be delegated to individual team members and developed in parallel, independent from each other. This assumes that the interfaces are well defined and the elements can be decoupled from one another.

Up to the preliminary design report most of the work was done in teams. The total weekly development time was approximately equal to the time the team spent together. Now team members can work in parallel or concurrently and the weekly development time is equal to the sum of the time of all members. This shortens the development cycle significantly. The team meeting agenda changes to coordination between team members, problem identification and solution, project planning and management.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.

- 1 slide with the functional architecture
- 1 slide with a schematic showing the major physical elements
- 1 slide depicting the major chunks with interfaces and fundamental interactions identified between components (additional slides to represent competing architectures if required—such as modular vs. integrated architectures)
- 1 slide showing the incidental interactions between the major chunks of the system (fundamental interactions may also be shown)
- 1-2 slides with a graphic depicting possible geometric arrangements of the major chunks

Component design specifications

Hardware Process

Overview

All products are made up of component parts which to a greater or lesser degree, have been defined in overall form during the conceptual design phase or during the product architecture definition. The physical (product) architecture defines the major chunks of the product and informs the requirements for the individual components that implement the functions within the chunks.

In order to proceed with the design of the product you must now improve your knowledge of these required component parts. You have to take into account the interfacing of the component with those adjacent to it and its effect on the total product. You have to decide on whether you want to make or buy the part.

In preparation for the detail design of the product, and also to discuss the component parts with potential sources of supply (e.g. vendors), it is useful to have a Component Design Specification (CDS). The major input to the CDS comes from the PDS and is in many cases identical. The main emphasis in the CDS is on the transformation of overall product specifications into the following:

- local performance
- local environment
- local constraints

Examples of typical CDS elements:

- Performance - input/output, load, maximum/minimum temperatures, voltages, currents, flow rates, etc.
- Local constraints - size, interfaces, interaction etc.
- Environment - temperature, humidity, corrosion, shock, etc.
- Testing
- Shelf life
- Quality, Reliability
- Weight
- Make or Buy - product sourcing
- Quantity
- Standard vs. Unique

- Cost
- Material
- Ergonomics/Safety
- Aesthetics

As with the PDS all CDS metrics should be quantified.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides summarizing list of unique & standard parts
- 1-2 slides with detailed specifications for 1-2 unique parts
- 1-2 slides with detailed specifications for 1-2 standard parts
- 1 slide with an example of the process used to determine make or buy for one component

Analytical and experimental plans/prototype test plan

Hardware **Software** **Process**

Overview

(See **Analytical and Experimental Report section**.) At this stage of the design process, at the latest, the team must develop a plan how to prove the concept and how to test the prototype or production sample against the PDS.

The analytical or experimental plan defines a structured approach applying modeling or statistical experimentation to ensure that design parameters are chosen properly and that the selected concept will work.

The objective of building a prototype or production sample is to verify the product design. The product design specifications are the criteria for verification. The test against these criteria is called the acceptance test. For example, if we translated the customer wants and needs correctly into product design specifications and if we verify our design against these specifications the product design can be accepted because it should meet the customer requirements.

Ideally the prototype consists of a complete product. In the case of larger products, components of the total product may be prototyped. Based on the results of building and testing the prototype, design modifications may be required.

While it seems early in the design process to think about the acceptance test it is important to plan now how we plan to verify the design, because it may influence the detail design, the timeline of the project plan and our resources. The team, therefore, should consider the following major points in preparing the analytical and experimental plans and the prototype test plan.

1. Develop detailed test objectives
    - What are the specific requirements to verify the design under test?
2. Develop/modify detailed test plans

- What are the planned test durations? e.g. time required to execute testing with and without problems. What are our problem projections?
- What will be the specific test vehicles to conduct the test? e.g. models, simulation, experiments, prototypes, production samples, etc.
- What are the specific resource requirements to complete the planned test activity?
- Should we plan for separate unit tests and subsequent system tests? e.g. develop and test of hardware parallel to software.

3. Detail the test design
   - Is our product design testable?
   - Do we have to change the product design to improve testability?
   - What techniques will maximize overall test efficiency and effectiveness? (design of experiments, guardbanding, etc.)
4. Prepare tests
   - Do we have the available resources such as personnel, tools and test equipment?
   - Are required tools and test equipment operational and calibrated?
   - Do we need special training?

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides summarizing analyses and experiments required for proof of concept (component and system-level)
- 1-2 slides summarizing verification tests required to prove the design meets the product meets specifications
- 1 slide special equipment, software, facilities, etc. required for testing
- 1 slide with the schedule

Preliminary manufacturing plan/product cost

Hardware

Overview

The objective of this deliverable is to establish a manufacturing plan which allows the design team to estimate the product costs. Major inputs into a total program cost estimate are:

- a summary of incurred and anticipated costs and expenses of HW/SW design
- activities (includes expenses for liaison engineer)
- expenses to release the product design to manufacturing
- manufacturing costs
- maintenance and/or support costs
- life cycle costs

The program cost estimate is the basis for your business case.

In this deliverable we want to concentrate only on the manufacturing cost e.g. what is the unit cost our sponsor company should expect. As you prepare for your System Level Design report, you will have to use these estimated manufacturing costs, update the other elements of the program cost estimate and prepare an updated business case.

To estimate manufacturing costs at this time you require inputs such as:

- product description
- product design assumptions
- assumptions where and how your intent to manufacture and test the product
- product sourcing assumptions e.g. whether you plan to make or buy the components of the product
- volume assumptions
- Bill of Material
- capital expenditures assumptions (tools, equipment, facilities)
- other assumptions such as
    - overhead
    - labor and burden rates
    - quality requirements
    - yields
    - capacity utilization (1 shift / 2 shifts / 3 shifts)
- *takt* time and line throughput
    - supplier quotes
    - competitive analysis, etc.

For future updates of the manufacturing cost, you should record and date the assumptions made.

In the manufacturing plan you should address where and how the product is being manufactured. To develop this plan you need essentially the same input assumptions as for the manufacturing costs. In addition you may have to consider:

- manufacturing location
- manufacturing line layout
- work in process and inventory
- production planning and control systems
- floor control systems, etc.

If these additional manufacturing plan assumptions impact manufacturing costs, you must include it in your cost estimate. Record and date all assumptions for future updates.

As you prepare the manufacturing cost estimate you may find that the product design to this point is not cost competitive. In that case, you have to initiate a cost reduction effort which can result in

- modification to the design
- different vendor sourcing
- different manufacturing process, etc.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides with bill of materials (identify sourcing & preliminary costs)--note that some projects may have a prototype BOM and a production BOM
- 1 slide with a summary of the product cost

- 1 slide with schedule of activities

System level design report

Hardware Software Process

Overview

(See **Major Deliverables** in IPPD Report Process section.)The System Level Design Report (SLDR) is a formalized, detailed technical specification of the product to this point. The level of detail may be project dependent but an attempt must be made to be as specific as possible. Insufficient details may impact the further development of the design resulting in potential project failure.

The SLDR must provide an overview of the overall design, design objectives and functional descriptions for Hardware (HW) and Software (SW). It must describe the major building blocks-subassemblies, SW subsystems and components. It must describe any operational and programming characteristics, interface specifications, operating environment and installation considerations. It must demonstrate that the selected design concepts have been verified, will meet the Product Design Specifications and can be translated into the detail design.

The overall SLDR should be completed at least 2 weeks prior to the formal System Level Design presentation to the sponsor companies in early December (see weekly schedule). The SLDR must be peer reviewed with the liaison engineer prior to the presentation. Issues must be formally recorded and individuals must be identified to resolve issues. One week should be allowed for issue resolution.

To proceed expeditiously with software development the SW part of the SLDR should be completed earlier. A target date should be 2 to 3 weeks after the due date of the preliminary design report (see weekly schedule in Canvas). This report should be peer reviewed with subsequent issue resolution. Proposed report content

Executive Summary (~1 page)

- Provides a quick overview of the report. The reader should gain a good idea of what the project was all about and what the results were. Key technical specifications, experimental and prototype results, and financial measures should be included.

Table of Contents

- Table of Contents
- Reference Documents
- List of Figures
- List of Tables

1.0 Introduction, Purpose, Scope of Report

2.0 Project Overview

- General Overview/Background
- Customer Needs
- Overall Product Design Specifications
- Product Architecture
- Project Plan
- Business Case

3.0 Hardware Design Description

- Overview/Objectives
- Hardware Architecture
- Organization and Operation
- Proof of Concept
    - analytical, physical analysis results (power, space, heat transfer etc.)
    - modeling results
    - design of experiments results
- Preliminary Drawings
    - 3D view
    - final assembly
    - subassemblies
    - special components
    - block diagrams
    - layouts
- Hardware Product Design Specifications
    - subsystems
    - components

4.0 Operational & Programming Characteristics

- Overview/Objectives
- Software Interfaces
- Memory Maps
- Microprocessors
- Registers, etc.

5.0 External Interface Specifications

6.0 Operating Environment

7.0 Installation Considerations

8.0 Software Design Description

- Overview/Objectives
- Software Architecture
- High Level Description of Each Subsystem
- Interfaces Between Subsystems

- Proof of Concept
    - reuse of code
    - new code
    - processor utilization
    - interrupt latency/servicing

9.0 Software Functional Description

- Configuration and Operation
- Error Handling

10.0 Software Interface Specifications

11.0 Material

- Overview/Objectives
- Preliminary Bill of Material
- Cost Estimate

12.0 Quality/Reliability Objectives and Plans

13.0 Manufacturing Plan

- Product Sourcing
- Assembly and Test Process
- Production Systems
- Tools and Equipment

14.0 Prototype Plan

- Sourcing (parts, assemblies, testing)
- Quantity
- Cost
- Schedule
    - parts procurement
    - sub and final assembly
    - test

15.0 Product Verification Plans

- Software Unit and System Test
- Hardware Unit and System Test
- HW/SW Integration Test
- Product Acceptance Test
- Regulatory Compliance Test

16.0 Issues, Concerns, Risks and Opportunities

17.0 Recommendations

Packaging the report: The System Level Design Report should be professionally labeled and packaged. It serves as the basis for a 20-minute oral presentation to the project sponsor company, an audience of other sponsors, other project teams and faculty from the University of Florida. The presentation will be peer reviewed before an audience of other project teams and faculty from the University of Florida prior the formal design review.

SLDR Report submission requirements

The System Level Design Report (SLDR) is a detailed specification of the project and serves as a crucial guide to the second semester detail design, implementation activities, and recommendations. An approval page, signed by the project team, faculty coach and liaison engineers is to be submitted with the report. The document is expected to be of professional quality. **You may find a Sample SLDR report in Canvas under Files in the "Deliverables" folder.**

Do not e-mail the report to Dr. Stanfill. Follow the instructions below.

Refer to the Signature Page Specification and Examples document in Canvas under Files in the "Deliverables" folder.

Report Evaluation Form

See the SLDR Rubric located in Canvas under Files in the "Deliverables" folder.

Submitting the report

1. Save the SLDR in Adobe Acrobat pdf format (design reports) and PowerPoint (presentations). The design station computers have Adobe Acrobat Professional. This application allows one to create pdf files via the MS Word, Excel, PowerPoint, Visio, etc. print menu. Acrobat Pro also allows multiple pdf documents to be merged into a single document (useful for compiling appendices).

Use the following naming convention for the documents:

teamname-SLDR.pdf

For example, team CPI will submit the following files:

CPI-SLDR.pdf - up-to-date system level design report

CPI-SLDR-AppX.pdf - appendices as required for the SDLR

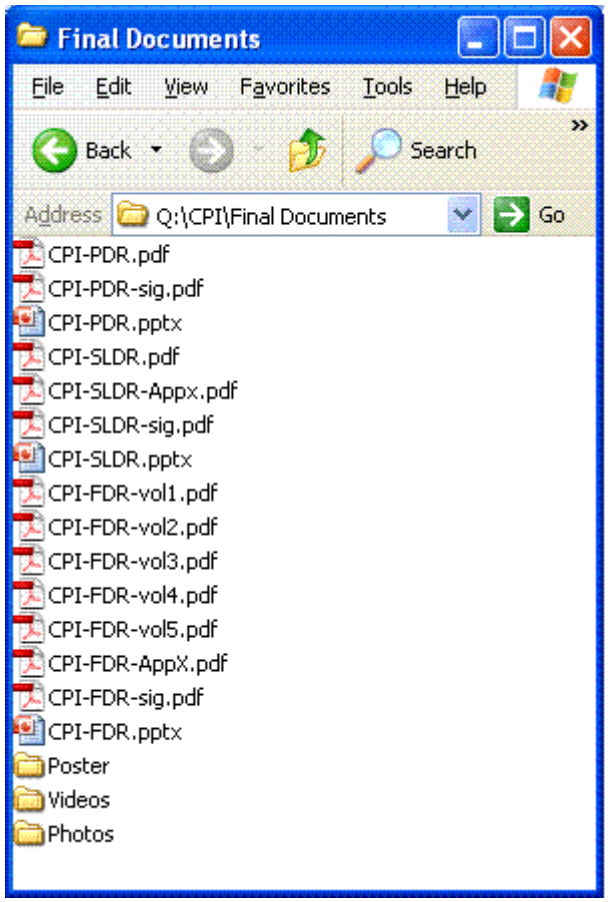
CPI-SLDR-sig.pdf - signature page for preliminary design report

CPI-SLDR.ppt - system level design review presentation

(be sure to include support materials such as videos)

- Put the files in the “final_documents” folder located in your team's repository.

Example:



- Please take this opportunity to upload your PDR report and presentation to the same folder, using the same naming convention.
- Your wiki site should also include the final version of these documents, including the original .doc, .ppt and .xls formats

Questions on how to access your team's IPPD account should be directed to support@ippd.ufl.edu

Review One-on-One with Dr. Stanfill

Each team will send two representatives to see Dr. Stanfill for a review of the SLD report. Be prepared to take notes and make corrections prior to last exam day. To schedule a time, visit the Doodle poll posted on the weekly schedule in Canvas.

Detail product and process design

Hardware Process

Detail design is essentially concerned with the design and manufacturing of subsystems and components that make up the whole design. As you proceed with the detail design you may have to expand the selected concepts into more detail to:

- better understand the concept
- sort out difficulties in the technological approach.

Major points to consider during detail design:

1. Always refer to the selected concept
2. Consider the interactions between the various areas of the design
3. Define component constraints to the system
4. Simple and cheap component design may not be the most economical for the whole of the project
5. Reduction of component variety generally leads to shorter development times and lower costs
6. Consider Design for X (DFx) during the design, e.g. Design for Manufacturing/Assembly/Test/ Cost/ Quality, etc.
7. Can you use standard parts?
8. Can the design be simpler?
9. Can parts or functions be split into simpler pieces?
10. Consider the tradeoff between the design of complex parts and available development time.

The objective of this deliverable is to provide a set of documents that contain all information necessary to manufacture the product. It includes part and assembly drawings, software programs, manufacturing process descriptions, vendor part information, assembly instructions, test procedures, proposed tool and test equipment and installation/maintenance procedures.

Analytical and experimental report

Hardware Process

Overview

The objective of this deliverable is to create a report summarizing the results of the analytical and experimental investigations that were defined in the **analytical and experimental plans/prototype test plan** deliverable defined during the System Level Design phase.

Proposed report content

Consider that this report should follow standards practices for a laboratory report. It is recommended that the analytical and experimental plans/prototype test plan be used as a starting point for this deliverable. This report should include, at a minimum, the following:

- table of contents
- detailed test procedures (the procedures defined in the test plan may not have provided adequate definition to complete each test or analysis, so be sure to include this new information)

- detailed list of materials, components, and equipment required
- photos and screenshots of the testing
- test set up diagrams
- summary of results
- statistical analysis of the data (such as ANOVA and t-tests)
- interpretation of the results
- recommendations
- appendix
    - raw data
    - detailed testing scripts and source code
    - detailed schematics
    - testing log entries for each experiment(including expected and actual results)

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide with a table listing each test
- 1-2 slides summarizing the results of each major test or analysis
- 1 slide with recommendations and next actions

Detailed manufacturing process and tool documentation

Hardware

Overview

This deliverable defines the work instructions (or operational method sheets) for each of the fabrication and assembly steps for creating the final product, plus the definition of all the tools, fixtures and jigs required for production. This deliverable does not include the layout of the facility. The facilities will be defined in the **final manufacturing and test plan** deliverable.

The work instructions provide step-by-step descriptions of how the product is made. Similar to a Bill of Materials, the work instructions should be organized into a Bill of Processes (BOP). The work instructions should be loaded with fully labeled photographs, figures and drawings depicting each process step. Key quality checks and specifications (dimensions, torques, voltages, colors, etc.) should be clearly flagged in the text and on the figures. Required tools, fixtures and jigs should be referenced for each step. It is very important to create a numbering scheme for the steps so that specific elements can be cross referenced and quickly located. To simplify the documentation of the work instructions, it may be useful to create a 2-column table with column one devoted to the labeled graphics, and column two devoted to the text-based instructions. Other documentation techniques include using spreadsheets such as Excel or presentation software such as PowerPoint to define the process steps.

The tool documentation should include a numbered list of each tool, fixture or jig required for fabrication, assembly and testing. For simplicity, tools, fixtures and jigs will be referred to as "tooling." All of the tooling should be numbered and organized into a Bill of Tools (BOT). In general, there is no need to provide great detail on common tools such as screw drivers, combination wrenches, and socket wrenches. For example, 12mm socket, 12 oz. ball peen hammer, and standard screwdrivers, would only need a one-line text-based description.

Commercial-Off-The-Shelf (COTS) elements should reference and included in the appendix. Links to specialty COTS tools should be provided for reference. Dimensioned drawings for custom fixtures and jigs should be included in the appendix.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide with a simplified Bill of Processes table
- 1-2 slides depicting a sample work instruction
- 1 slide with a simplified Bill of Tools table
- 1-2 slides showing examples of custom tooling

Acceptance test results and report

Hardware Software Process

Overview

The Acceptance test results and report summarizes the tests conducted to prove the product or process met all the product design specifications that were agreed upon for the project. The report includes interpretations of the results and recommendations for follow-on actions required to achieve unmet specifications.

Proposed report content

The report should include a summary table showing the mapping between product specifications (with target), tests conducted, and results (it is acceptable for the results to be "pass" or "fail," or "met" or "unmet"). The Summaries of each test, including an overview of the procedure followed (with test vehicles, schematics, photos, etc.), a data summary, an interpretation of the results, and the recommendations should follow the summary table. A section entitled summary and recommendations should conclude the main report. An appendix with the following content completes the report:

- details of each test
- test logs
- raw data

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides showing the test matrix
- 1-2 slides with showing the detailed summary of one test along with recommendations
- 1-2 slides with recommendations

Final manufacturing and test plan

Hardware

(Go back to **Detailed manufacturing process and tool documentation**)

Overview

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides with

Final product cost

Hardware

Overview

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides with

Final report and project documentation

Hardware Software Process

Overview

(See **Project Roadmap** section.) The final report and project documentation contains all information about the project. It includes product and process descriptions, the results of the acceptance test, product manuals and any other important information about the project. The documentation should be complete so that it could be used to get final bids on producing the product and to use and maintain the product. The exact content of the project documentation may vary and should be determined together with the faculty coach and the liaison engineer.

The project documentation will consist of a number of volumes. The first volume is the final report and refers to the other volumes of the documentation. The specific content of each volume is described later.

An oral presentation of 20 minutes will be made to the project sponsor company an audience of other sponsors, other project teams, and faculty from the University of Florida. The objective of this presentation is to show the sponsor the result of the project. If practical a production sample, a model or a photograph of the project will be displayed.

Documentation requirements

The project documentation consists of six volumes. The first four volumes (copies) and volume 6, the notebooks (if requested) will be delivered to the sponsor company. The first five volumes (originals) will be delivered to IPPD and archived at the University of Florida, College of Engineering. The six volumes are:

Volume 1: Final Report

Volume 2: Product and Process Documentation

Volume 3: Acceptance Test Results and Report

Volume 4: Product manuals

Volume 5: UP-TO-DATE Deliverables of entire Project

Volume 6: Design Notebooks

Volume 7: Prototype Demonstration Video

Each volume should have a cover sheet with the IPPD logo, the project logo, the volume number, date, name of the project and the sponsor company (except design notebooks). The entire project documentation should be bound for submittal to the sponsor company and one copy in electronic format for the College of Engineering. The content of each volume is described below:

First report element: Master Table of Contents

An outline of the final report with reference to volumes 2, 3 and 4, including a brief description of what is in each volume.

Final Design Report

(See **Major Deliverables** in IPPD Report Process section.) This report should be brief and to the point, assuming that the audience of the report is the project sponsor company management. The final report can serve as a structure for the final oral presentation.

Proposed outline:

Executive Summary (~1 page)

- Provides a quick overview of the report. The reader should gain a good idea of what the project was all about and what the results were. Key technical specifications, experimental and prototype results, and financial measures should be included. An executive summary should document the key requirements and specifications, highlight the selected concept, outline major project

milestones, identify major risks and opportunities, describe major test results, and summarize the business case and recommendations. The executive summary must include quantitative information.

- *This one-page (if single-spaced) section should follow the report title page and **precede** the table of contents.*

Table of Contents (~2 pages)

- Table of Contents
- Reference Documents
- List of Figures
- List of Tables

1.0 Introduction (~1 page)

- Focuses on the problem description and a brief background of the customer and its needs.

2.0 Main Part of the Report (~8 pages)

- Describes the results and how they were obtained. References to product specifications, acceptance test results and other volumes. Uses figures and pictures.

3.0 Conclusion (~1 page)

4.0 Appendix

- At a minimum, the appendix should include a process assessment of the proposed project roadmap and the actual process followed. This documentation will consist of a copy of the proposed project roadmap, the actual roadmap process followed, and an analysis of each deliverable completed during execution of project. The analysis should include a candid review of what was and was not beneficial to the final results of the project and suggested improvements for the deliverable. If a new deliverable was created, then it is very important to document the recommended content and the timing.

Due Dates

Due Date	**Description**
Mid-Apr.	Complete Final Design Report and review final, signed-off version with Dr. Stanfill. Please, check with your coaches to find out more.
End-Apr.	Completed final checklist with all approval signatures turned in to Dr. Stanfill, 378 Weil. See the **Final Checklist in the weekly schedule and deliverables link in Canvas.**

Meet with Dr. Stanfill

Each team needs to have one to two representatives meet with Dr. Stanfill to review the submission of your final report and the **FINAL CHECKLIST (found in Canvas)**. Please visit the **DOODLE poll (posted in Weekly Schedule on Canvas)** and select a meeting time. **Please be sure to have the signatures from the Lab Manager and your coach prior to visiting with Dr. Stanfill**. Note that the report must be of professional quality and all the items completed before grades will be released.

Do not e-mail the report to Dr. Stanfill. Follow the instructions below.

Signature page information

Please refer to the Signature Page Specification and Examples PDF located in Canvas in the Files folder under the "Deliverables" tab.

Submitting the report

1. Save the FDR in Adobe Acrobat pdf format (design reports) and PowerPoint (presentations). The design station computers have Adobe Acrobat Professional. This application allows one to create pdf files via the MS Word, Excel, PowerPoint, Visio, etc. print menu. Acrobat Pro also allows multiple pdf documents to be merged into a single document (useful for compiling appendices).

Use the following naming convention for the documents:

teamname-FDR-vol#.pdf

For example, team CPI will submit the following files:

CPI-PDR.pdf - up-to-date preliminary design report

CPI-SLDR-sig.pdf - signature page for preliminary design report

CPI-PDR.pptx - preliminary design review presentation

CPI-SLDR.pdf - up-to-date system level design report

CPI-SLDR-AppX.pdf - appendices as required for the SLDR

CPI-SLDR-sig.pdf - signature page for system level design report

CPI-SLDR.pptx - system level design review presentation

CPI-FDR-vol1.pdf - final design report volume 1

CPI-FDR-vol2.pdf - final design report volume 2

CPI-FDR-vol3.pdf - final design report volume 3

CPI-FDR-vol4.pdf - final design report volume 4

CPI-FDR-vol5.pdf - final design report volume 5

CPI-FDR-AppX.pdf - appendices as required for the FDR

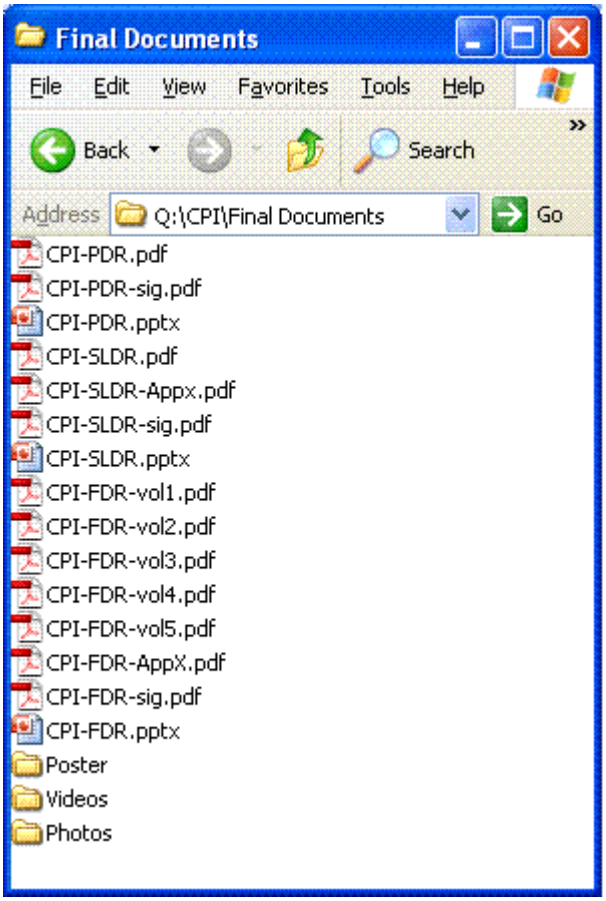
CPI-FDR-sig.pdf - signature page for final design report

CPI-FDR.pptx - final design review presentation

/Poster/ - folder with poster files
/Videos/ - folder with videos
/Photos/ - folder with prototype photos
/Original format/ - folder with docx versions of report volumes

- Put the files in the "Final Documents" folder located in your team's repository. Be sure to synch with the IPPD server prior to meeting with Dr. Stanfill. Each member of the team has access to this folder.

Example:



- Your wiki site should also include links to the final versions of these documents, including the original .doc, .ppt and .xls formats

Questions on how to access your team's SVN repository should be directed to support@ippd.ufl.edu

Product and process documentation

The first page provides a table of contents with page numbers. Includes all pertinent information to describe the product and processes, such as:

- product and subsystems/component specifications
- part and assembly drawings
- SW programs
- circuit diagrams
- bill of materials - parts list
- process specifications
- process instructions
- assembly instructions
- test procedures, etc.

Includes other important product information such as:

- quality plans
- manufacturing plans (sourcing, tools, equipment. etc.)
- cost estimates
- financial analysis
- product safety assessment
- legal evaluation
- environmental impact assessment, etc.

Acceptance test results and report

The first page provides a table of content with page numbers. Describes test vehicle, test requirements, test methodology, test environment, etc. and the test results.

Product manuals

The first page provides a table of content with page numbers. Includes information about user interfaces, maintenance, installation, calibration, etc. The content of this volume may vary by project.

Deliverables

The first page provides a table of content with page numbers. This volume should include the latest updates of the following major deliverables:

- concept generation and evaluation
- conceptual design report
- project plan
- system level design report
- analytical and experimental plan and report
- prototype results and report (include photos if possible)

Design notebooks

The sponsor may wish to keep the design notebooks. At the end of the project, turn in your design notebook to your coach.

Prototype demonstration video

The prototype demonstration video is a project artifact that is intended to serve several purposes. First, the video illustrates the use of the product under controlled conditions that may not be practical to recreate during a live demonstration in an office or classroom environment. For instance, the size of the prototype, special power, hardware, software or network requirements, available only in a laboratory or industrial setting, may not be feasible for a live exhibition. Actual usage may require more time than can be allotted for in a presentation and the video allows for editing out long wait times between operations (such as heating, cooling, and drying).

Second, the video also provides a way to demonstrate a variety of use cases for the system. These uses may include installation, configuration, maintenance and emergency operations. For a software application, this demonstration may serve as a user tutorial.

Third, the video concisely describes the project and the capabilities of the design team. The video should provide a sense of the typical project scope and expected deliverables. This aspect is useful for informing either potential new clients or attracting new engineers interested in joining the IPPD program.

Lastly, the video serves as a tool to inspire future design teams.

Each project will include two versions of the video. The first version will be a 20-second trailer, that may be included within a PowerPoint presentation. The second video will be a full length video, that should be no longer than about 2 minutes.

More information on the video is available in **Appendix: Project Video Requirements**.

IPPD Software Deliverables

Introduction

This chapter is being consolidated into the previous chapter. Sections that have been incorporated will be ~~lined out.~~

Overview

Back to Intermediate Deliverables

In the Software Development Process, we have chosen a set of major deliverables that are generally applicable. They will allow us to monitor the progress of a project, to evaluate the student team and to report the status of the project to the industry sponsor.

Generally these deliverables are almost always appropriate. In certain circumstances details of a deliverable have to be customized to reflect the specifics of a particular project. Deliverable deadlines may not be changed. Specific deliverable content may be changed if absolutely necessary.

As discussed in the Software projects section, agile development methodologies, such as SCRUM, may be used in lieu of the structured development process described in this chapter. Your project coach can assist you in determining how to adapt the content and timing of the major software deliverables described in the following sections to your particular project.

~~Technical strategy~~

~~Overview~~

~~Requirements for a product must be evaluated in the context of a technical strategy that provides the strategy direction for the product. The technical strategy should describe:~~

- ~~the basic structure and direction of the product~~
- ~~the relationship of the product to any pertinent architectural standards~~
- ~~user interface standards~~
- ~~interface standards with other products~~
- ~~direction for modification or replacement of old source code~~
- ~~the software engineering environment and tools required for the product such as:~~
  - ~~program language~~
  - ~~compilers~~
  - ~~equipment and operating systems~~

~~The technical strategy does not describe the internal design or implementation of the product.~~

~~Presenting this deliverable~~

- ~~1 slide with team name, logo, team members, etc.~~
- ~~1-2 slides with basic structure & product direction; a list of pertinent architectural, user interface standards, and system interface standards~~
- ~~1 slide with targets for requirements, spec and code reuse~~

- 1 slide with list of software environment, tools, programming environment, compliers, equipment & OS (mention configuration management system if known)

System/product requirements

Overview

Determining requirements or customer needs for a product can be challenging because requirements are not always known with precision at the outset and also because they may change over time.

The objective of this deliverable is to determine requirements to be met by the design of the product. Requirements should be unambiguous, validated and prioritized by the customer of the product. Once requirements have been determined they need to be baselined i.e. stored in a requirements database.

The following is a proposed four-step process to determine requirements.

Collecting input requirements: Each input requirement is numbered and includes a brief description of attributes related to function. Customers of the product are generally the source of input requirements. Market research, literature and competitive analysis may be other resources.

Problem analysis: This analysis should provide a detail description of each underlying problem from an end user prospective. It contains information needed to understand the requirement. In addition to control information it may contain a list of problem conditions, descriptions of current functions, user tasks, end-users and the environment. Analyzing problems can be an iterative process until it is agreed that the problem description is complete, consistent and correct.

Solution analysis & definition: The purpose of solution analysis and definition is to develop one or more effective, external solutions for each selected problem definition. It may include control information, proposed function, resolved problem conditions, modified or new user tasks, affected environment and end-user, and a value assessment for the particular solution.

Programming objectives: Programming objectives incorporate a set of solution definitions into a coordinated description of new and enhanced functions. As programming objectives are being developed the level of solutions as previously defined may have to be refined. This is an iterative process until the Programming Objectives satisfy the Problem Definitions.

System modeling can be a part of requirements definition. Entity relationship diagrams (ERD) would be one example of data modeling aspect of a system model. Viewpoint or data flow models might be another.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-3 slides with a numbered list of requirements (operational characteristics)

- 1 slide with an example of problem analysis, solution analysis & definition for a sample requirement
- 1 slide with sample programming objectives for solution definitions, or an appropriate system model such as an ERD, dataflow or viewpoint model.

Complete and testable product specifications

Overview

(See **Intermediate Deliverables** section.) **Requirements and specifications are not the same:** Sometimes software requirements and software specifications are used synonymously. In the IPPD software development process we want to distinguish requirements and specifications.

Requirements are generally a qualitative description of customer needs. For example a customer need may be the following:

"The software shall provide real-time response to sales activity queries."

Specifications are a quantitative description of the requirements and must be measurable and testable. The above customer need can be good to determine programming objectives but it is not precise enough to test it.

The more testable version of this requirement, called the specification, could be:

Type A queries in less than .5 sec.

Type B queries in less than 1 sec.

Type C queries in less than 2 sec.

Specifications must be testable to the extent that one can define a clear pass/fail test for determining whether or not the developed software will satisfy the specifications. In order to be testable, specifications must be specific, unambiguous and quantitative wherever possible. All specifications must be numbers - especially for reference in test documents.

Example: Specifications 16 and 17

<S-16>	The product shall be capable of supporting data signaling transfer among modules in a full duplex rate from 56 Kbps to 4 Mbs.
<S-17>	The product shall support a time and date clock. Time and date shall be retained across power cycles.

It may require significant effort to eliminate the vagueness and ambiguity in a specification and make it testable. It may require iteration (i.e. rapid prototyping) between requirements and

specification to achieve affordable testability. Determining testable specifications at this stage is also important in preparation for a product verification strategy.

Product/system requirements and product specifications checklist:

12. Is the requirements document complete, i.e., does it reflect all of the known customer needs?
13. Are the specifications so detailed that designing and coding are restricted to a single implementation?
14. Does the human interface follow established standards?
15. Has the entire infrastructure been specified, i.e., backup, recovery, initialization, etc.
16. Have all reliability and performance issues been listed?
17. Have all security considerations been listed?
18. Do the requirements consider all existing constraints?
19. Do the requirements provide an adequate base for design?
20. Are the requirements complete, correct, and unambiguous?
21. Are all specifications testable?
22. Are all interface requirements feasible?

Software development should only proceed after satisfactory answers to the questions in this checklist have been provided or actions to resolve outstanding issues have been defined with a date for resolution.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 to n slides with a numbered list of software specifications
- 1 to 2 slides with examples of how specifications will be tested

System/product architecture

Overview

The objective of this deliverable is to provide a clear overall system/ product organization including a good architectural overview and justification. The total system/product is broken down into well-defined modules including their functionality and interfaces to each other. All functions listed in the requirements must be covered neither by too many nor too few modules. The architecture should be designed to accommodate likely changes within a module without affecting the rest of the modules. Major data structures need to be described and justified. Database organization and content must be specified.

A clear architecture with well-defined modules allows delegation of specific modules to development engineers for parallel or concurrent development of a product.

System/product architecture checklist:

10. Is the overall program organization clear, including a good architectural overview and justification?

11. Are the components (modules and objectives) well defined including their functionality and interfaces to other components?
12. Are all the functions that are listed in the requirements covered?
13. Is the architecture designed to accommodate likely changes?
14. Are all major data structures described and justified?
15. Is the database organization and content specified?
16. Have all the dependencies been identified?
17. Is the user interface modularized so that changes in it won't affect the rest of the program?
18. Are key aspects of the user interface defined?

Software development should only proceed after satisfactory answers to the questions in this checklist have been provided or actions to resolve outstanding issues have been defined with a date for resolution.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1 slide with a graphic showing where system/product fits within existing environment – identifying all interfaces
- 1 slide with system/product broken down into well-defined modules with interfaces between modules (UML or ERD) and functions each module supports
- 1 slide describing major data structures
- 1 slide with sample user interface

Prototype plans

Overview

Through prototyping the functional behavior of a system/product or module can be observed early in the development cycle. Rapid prototyping and evolutionary design can be an alternative to the generation and validation of written requirements/specifications as a way of ensuring that a software product will be responsive to user needs. Major advantages that a prototype can provide include the following:

- a mechanism that allows designers to evaluate alternatives and to investigate problems
- an early assessment of the impact of a new function to reduce risk
- evaluation of changing environmental characteristics
- improved requirements, specifications, design verifications and user interfaces

Prototype plans should be made early in the development phase and should be determined based on guidelines such as the following:

- What do we need to know about user interfaces (screens, screen transitions, dialogs, reports etc.)?
- What is our plan to evaluate the prototype feedback?
- What do we need to know about data flow, internal interfaces, and feasibility of design or performance issues?

~~Prototyping results can be used to make a better product. Generally, however, prototype code should not be used in the formal product. A prototype by design and intent answers specific questions. Attempting to implement the prototype as the operational product can result in new problems as new questions surface.~~

~~Things to avoid when developing a prototype:~~

- ~~rushing into prototyping before adequately defining technical objectives~~
- ~~losing sight of prototyping objectives~~
- ~~developing the prototype into the product~~
- ~~making endless prototype iterations~~
- ~~delaying documenting the prototype results~~

~~Presenting this deliverable~~

- ~~1 slide with team name, logo, team members, etc.~~
- ~~1 slide with specific questions trying to be answered by one or more software prototypes~~
- ~~1 slide with list of prototypes, tool used for prototyping (i.e. a 4GL, Excel macro, VB application), resources and schedule~~
- ~~1 slide with prototype evaluation criteria (how to judge results & what to do with the info)~~

Preliminary design report

- An outline for the

Preliminary design report and project plan major deliverable is provided in the

Comprehensive test planning

Comprehensive test plan

The Comprehensive Test Plan (CTP) is a master plan, which defines all design evaluation activities required for the product including its components to assure that the product will perform as specified. Comprehensive means that its scope will include all design verification, simulation and test activities from concept through final product.

CTP major objectives:

- identify all design verification, simulation and test activities
- integrate the hierarchy of verification and test plans from module to integrated system
- identify resources and schedules for the proposed tests
- determine entry and exit criteria for each phase
- identify the test environment, test tools & equipment, test simulations & drivers and test stations
- identify and eliminate redundancies in the development cycle related to design verification and testing
- provide a control vehicle to measure progress

The Comprehensive Test Plan is an evolving plan, becoming more detailed throughout the product development cycle. The CTP should provide a product testing strategy rather than detailed test procedures and detailed test requirements.

Depending on the complexity and size of a product there may be up to four different test stages in the hierarchy of the Comprehensive Test Plan.

Unit test plan

The major objective of a unit test is initial verification of the proper execution of all new, changed and affected paths in each module. This can be done with or sometimes without a fully built driver. Unit testing technically occurs before code is integrated into the change control database.

Unit test is the initial testing of new and changed code and interfaces in a module. It should verify the following:

- all branches execute properly in each module
- implementation at the module level is defect free
- error recovery works properly at the module level

Functional verification test plan

This test verifies proper execution of product components. It does not require that the product being tested interface with other products. Simulation can be used to create an acceptable test environment because often schedules require that one product enter testing before others are ready. Also it is more effective to carry out initial testing in a controlled and isolated test environment.

Purpose:

- Exercise new, changed and affected product functions against programming specifications.
- Exercise new, changed and affected interfaces between modules and components and verify that they function correctly.

Product verification test plan

This test verifies the entire product including usability, performance, installability and the error paths. Usually this test requires a direct interface with other software products.

Purpose:

- Exercise all product functions and verify proper execution with other software products.
- Verify usability, performance and installability characteristics.

System verification test plan

This test verifies the proper execution of all related software and hardware products. It is performed using only the actual software and hardware without any simulators. This test simulates the user environment and emphasizes stress, performance, security and cross-product interaction.

The System Verification Test is the final test stage.

Purpose:

- Execute the product in a system environment representing the customer environment as physically and financially practical.
- Verify the quality of the product to include functional adherence to the product specifications and all usability, reliability, recoverability, performance, installability and operational characteristics.

Presenting this deliverable

- 1 slide with team name, logo, team members, etc.
- 1-2 slides summarizing the product testing strategy, the testing hierarchy required (could including all of the following testing plans: unit, functional verification,
- product verification & system verification), and the entry and exit criteria for each phase
- 1 slide identifying the test environments, tools & equipment, test simulations & drivers, and test stations
- 1 slide with an example test for each phase
- 1 slide with the schedule and resource assignments

Software development methodologies

The following major software development methodologies are not stand-alone deliverables but need to be part of any previously described deliverables.

~~Configuration management plan~~

~~Configuration management is a process of identifying the configuration (parts) of a product for the purpose of systematically controlling changes to the part.~~

~~The configuration management plan defines the goals, structure, responsibilities and degree of change control. The plan is created during Phase I of the IPPD Software Development process and defines the following:~~

- ~~the baselines to be established during the development cycle~~
- ~~the objectives to be included in each baseline~~
- ~~database, library structure, responsibilities and tools to be used for managing changes~~
- ~~a set of change control criteria for~~

- level / method of authorization
- type of verification

- change control criteria such as
  - changes of the design or specifications
  - cause of change (errors, enhancements, requirement changes)
  - impact on lines of code, documentation, work items, personal hours

Baselines include those work items that have been verified and agreed upon to serve as the basis for further development. Once the basis is established no work item may be added or modified without appropriate authorization.

Presenting this deliverable:

- 1 slide with team name, logo, team members, etc.
- 1 slide identifying baselines for software builds & objectives for each build
- 1 slide with change control criteria (numbering & naming conventions)
- 1 slide enumerating what objects under change control
- (i.e. requirements, specs, architecture, code, test plan, etc.)

Reviews, inspections and walkthroughs

Reviews or inspections and walkthroughs are validation procedures. They are used for verifying each work item at least once before it is included in the product.

Reviews/inspections: Involve a period of intense study and comparison of a work item against the reviewer's standard of comparison or previous related items. For example: Review of the Technical Design Specification by the liaison engineer during November. Issues are identified, recorded and assigned to an individual for resolution by a determined date.

Walkthrough: A simulation event conducted by a group of trained professionals. Participants who individually keep track of concurrent states of the walkthrough object play roles and report observed problems.

For example a usability walkthrough verifies the product design meets what the user wants. During such a walkthrough product developers attempt to view their design from a user perspective by role-playing.

In-process measurements

The IPPD Software Development cycle is only eight months. To manage the tight schedule and keep track of progress made, a few metrics should be gathered, maintained and used to measure plan vs. actual and take corrective actions. These metrics represent a very small sample of what is generally used in industry.

Software requirements: Capture of the total number of requirements for the product. Plot of the number of changes each month. This metric indicates the stability or volatility of the baseline which correlates to the schedule performance.

Software size: Planned vs. actual size of the product in terms of lines of code (new, used, modified).

Software coded and unit tested: Number of software modules planned vs. completed by month.

Software defects: Defects per 1000 line of codes due to defective coding, requirements, design decisions, interface, etc.

Peer, Faculty, and Sponsor Reviews

Overview

(See **Preliminary Design Report Peer Review** section.) Project reviews are key activities associated with effective product and process design. The project review process allows project stakeholders not intimately familiar with the project particulars to contribute to project through a process of questioning and providing constructive feedback.

How many slides are appropriate? If you consider that 40 seconds per slide is average, then for a 15-minute presentation, 23 slides is about right. Guy Kawasaki's formula for winning presentations follows the *30-20-10 rule*: 30-point font (minimum), 20-minute talk, 10 slides. It is very difficult to cover the required material and follow Guy's formula. Be sure that the audience understands the context for your presentation--why should they care? who is your customer? what does the customer want? what do you propose? how will you accomplish it? what will it cost? what are the issues? what are your recommendations?

Important instructional and schedule documents can be found in Canvas in the weekly schedule topic links.

Practice for success: It is important for project team members to be able to communicate technical and business project information to a variety of technical audiences. Further, it is necessary for all project team members to gain experience and confidence in concisely delivering a technical presentation. The keys to improving presentation techniques are to practice as an individual, practice as a team, and practice in front of an audience of your peers.

As mentioned earlier in this guide, the following three types of scheduled reviews are used in the IPPD process:

1. peer reviews with other IPPD teams
2. design reviews with sponsor companies
3. progress reviews with a panel of faculty coaches

The peer reviews provide a venue for teams to deliver a 15-minute dry run of their design review presentation to an audience of their fellow engineers. These reviews are held prior to the formal presentations to the sponsor companies. The peer reviews allow the teams to receive feedback from project coaches and classmates, and to practice handling questions from the audience. Presentation content, configuration, and delivery are scrutinized during these sessions.

The sponsor company design reviews may be held on campus, via tele/video/web conferencing, or live at the sponsor's site. These sessions run anywhere from one hour to half a day. Typically, the liaison engineer will invite fellow engineers and managers to these review sessions.

The progress reviews with faculty are held in private conference rooms and are limited to 30 minutes. These reviews are used to identify risks, issues, and action plans to insure projects are delivered on time and within budget.

Peer reviews with other teams

Peer reviews with other teams are held prior to the sponsor review. Each presentation room with host three to five project teams and their faculty coaches. Participants are required to stay for the entire session so that all teams will have an audience. The process is as follows:

- teams arrive early to preload their PowerPoint presentations onto the desktop of room's computer
- the faculty room coordinator passes out feedback forms to the audience
- teams present for 15 minutes and take questions/receive feedback for 5 minutes
- audience passes feedback forms to the room coordinator at the conclusion of each presentation
- repeat until all teams have presented

Important:	Each team should provide a unique, meaningful filename for their presentation. As a default, use <teamname>-PDR.ppt for the Preliminary Design Review
	Don't assume the room computer has access to the internet--bring your presentation on USB media
	Rehearse your presentation individually and as a team prior to the peer review
	Review the presentation with your liaison prior to the peer review. There may be some sensitive technology that the sponsor may not wish to reveal to the public

Preliminary Design Peer Review

The Preliminary Design Review (PDR) peer review presentation is typically an abbreviated version. Since the presentation is limited to 15 minutes, it is recommended that some content be shortened. For instance, you may wish to present the details of one concept instead of multiple concepts. You may also limit the number of issues or risks that you discuss.

System Level Design Peer Review

The System Level Design Review (SLDR) is held on the UF campus and is limited to 20 minutes per team, including questions. Therefore, the peer review presentation should be limited to 15 minutes.

Final Design Peer Review

The Final Design Review (FDR) is held on the UF campus and is limited to 20 minutes per team, including questions. Therefore, the peer review presentation should be limited to 15 minutes. It is likely that a longer, more detailed presentation will be delivered by the team at the sponsor's site, so like the PDR peer, be prepared to remove some content.

Reviews with sponsor companies

Preliminary Design Review

This review is typically held at the sponsor company's site. This allows the sponsor to provide a broader technical audience for the review. The intent of this review is to reach agreement with

the sponsor that the solution path chosen by the project team is technically feasible and appropriate, that the schedule is realistic, and that the project economics are satisfactory.

The team should plan to present at least thirty minutes of material and be ready to take lots of notes. Frequently the sponsor requires changes to project prior to signing off on the project proposal.

Preparation

It is important to send an electronic copy of the presentation ahead of your arrival. Ideally, your liaison engineer has already reviewed your presentation and has helped you prepare. In addition, the latest draft of the report should be sent to the liaison. Ask your liaison engineer to reserve a conference room and invite company representatives. Get a list of attendees and their titles ahead of time if possible. Be sure to find out what type of computer and multimedia facilities are available at the site. Do not forget to practice! Leave plenty of travel time in case unexpected delays occur in route. Plan to get there early.

What to bring

1. Find out if the sponsor company wants you to bring hardcopies of the design report
2. Back-up transparencies of the presentation
3. Hardcopies of the presentation for handouts (make sure you bring enough)
4. Electronic version of your presentation in CD-ROM and USB thumb drive formats
5. A laptop with the presentation preloaded

Upon arrival

1. Get to the presentation room as soon as practical
2. Set up the required equipment and make sure the presentation operates as designed
3. Go through one dry run with the liaison engineer as the audience
4. Now relax!

During the review

1. Assign one or more official scribes (everyone else should take notes, too)
2. Be sure to write down any action items and issues
3. At the conclusion of the review, be sure to read aloud all the action items and issues; make sure follow-up responsibilities and due dates are clear and agreed to

Follow-up activities

1. publish a design review meeting summary
2. close out open action items

System Level Design Review

The SLDR is held on the UF campus in conference-style format. All the sponsors, plus UF deans, department chairs, faculty, advisors, and students are invited to attend. The review features a breakfast networking session, 20-minute team presentations in three or more room simultaneously and a banquet luncheon with a keynote speaker. Dress code for the SLDR is business formal. The review is held on the first reading day prior to final exams in December.

Final Design Review

The FDR is held on the UF campus in conference-style format. All the sponsors, plus UF deans, department chairs, faculty, advisors, and students are invited to attend. The review features a breakfast networking session, 20-minute team presentations in three or more room simultaneously, a poster and prototype demonstration session, and a banquet luncheon with a keynote speaker. Dress for the FDR is business formal. The review is held about two to three weeks prior to final exams.

Qualification Review Boards

(See **Overview: Project Roadmap.**) Qualification Review Boards (QRBs) are special project reviews held before faculty expert panels. The QRBs occur in the second half of the IPPD program. The QRBs consist of project coaches with subject matter expertise in particular aspects of your project. The panel is responsible for helping each project team identify and mitigate project risks. For the process to work, it requires that the project teams openly share project risks, and be open to constructive criticism.

The reviews last thirty minutes and are held in private conference rooms. Teams should arrive ten minutes early and wait quietly in the hall outside the conference room.

Important:

- Use PowerPoint for these presentations
- Reuse the back-up slides from the SLDR and previous deliverables
- Prepare hardcopies of the presentation for the review committee (print the slides 2 or 3 to a page)--including the MS Project Gantt chart
- Assign one or more note takers

Project Plan Review Session

The purpose of the project plan review session is to determine if your project is on track, to identify weaknesses and to recommend corrective actions. The following items and questions will be covered during a detailed review of your project status. The project review will last 30 minutes. Plan on speaking to the following items for at least 20 minutes:

- Concise summary of project including key requirements, proposed solution and major scope changes
- Time line of major activities in the future (through April)

- Status of proof of concept (including software prototypes), experiments, Design of Experiments
- Status of Detail Design (Hardware and Software)
- Status of prototype:
    - Where will it be manufactured?
    - Who will manufacture it?
    - Who will fund it?
    - When will it be manufactured? (time line)
    - Where are we going to assemble and test it?
    - Are parts on order? When will they be ordered?
    - Will parts be available on time? Do we have commitments?
- Prototype (Hardware & Software) Test Plan
    - What are our objectives?
    - Do we have a (comprehensive test) plan?
    - Where will it be tested?
    - What equipment do we require?
    - How are we going to test against product specifications?
    - Are we trained to test it? Do we need special training?
- Risks and risk mitigation strategy
- Resource limitations
- Liaison engineer support and frequency of contact

Provide the reviewers with softcopies via e-mail at least four hours prior to the presentation.

Prototype Review Session

The purpose of the prototype results and report review session is to identify weaknesses in your project and to recommend corrective actions. It is vital to openly share project issues and identify where assistance is needed. The review will last 30 minutes; however, only the first five minutes will be in a predetermined order. The review panel will determine what happens after the initial "canned" content presentation.

Prepare a concise, five-minute presentation, to be delivered by no more than two speakers, that includes the following:

1. One-minute project overview (elevator speech) with key project requirements, proposed solution, and major scope changes
2. An architectural slide of the proposed system/solution that illustrates key interactions among the project chunks; for process-oriented projects, provide a flow chart with the main process elements
3. A Pert chart with major project activities and timing. This chart should illustrate the dependencies among the tasks. Either use MS Project (select the Network view), Visio or PowerPoint to create this graphic. Be sure to indicate the task owner, duration, and status
4. A risk chart consisting of a numbered list of project risks with assessment of probability of occurrence (i.e. high/low), negative impact potential (i.e. high/low), mitigation strategy, and owner

Back-up slides: Prepare back-up slides and organize the slides for ease of access. Provide the reviewers with softcopies via e-mail at least four hours prior to the presentation.

The back-up slides should include, but not be limited to the following:

- Status and results of prototype
    - Fabrication
    - Assembly
    - Coding
    - Testing
- Design of experiments results
- Schematics, detail drawings, models
- Comprehensive test plan (CTP) status
- Preparations for, or results of, product acceptance testing
- Status of project documentation
- Status of the project demonstration to be delivered at the final design review

Expectations for Teams and Individuals

Team expectations

Overview

There are two main goals to the Integrated Product and Process Design program. The most important is to teach new engineers a successful integrated product and process development process. The second goal is to have engineering teams successfully complete an engineering project for their industrial customer. Both goals are complementary and can be achieved if the team works well together.

To achieve the second goal, the team must meet customer needs on time and within budget. Meeting customer needs means that the team effectively identifies customer needs through consistent, regular interaction with the liaison engineer of the industrial customer. Meeting the needs on time means that the team must work effectively within the project schedule. Project deliverables must be ready on time and should be of highest quality. Working within budget means that the team will regularly monitor its spending.

Each project is scoped at 800 to 1000 hours for an experienced engineer. Assuming a team of five, each team member is expected to spend approximately 10 to 15 hours on the project per week. This time is in addition to the weekly general lectures and the formally scheduled weekly project specific workshop.

Team meetings

Time together as a team is at a premium and should not be wasted. Team meetings should follow accepted practices for effective meetings. This includes the preparation of an agenda ideally distributed to team members before the meeting; keeping minutes and timely distribution of meeting minutes, including assignments made and conducting the meeting.

To conduct their business it is expected that teams must meet at least weekly. Faculty coaches should be invited as required. Minutes of the meeting should be provided to the coach for meetings where the coach is not included.

Team selection

A critical issue in the performance of an engineering project is the dynamics of the team. IPPD is committed to making sure all teams are successful. Assignments to teams are done using as much information as possible about potential team members. Following are some of the criteria considered:

- Expertise Area of team members. Considering preliminary information about the industry design projects we determined how many students from each discipline would be required per project.
- Personality of team members based on input from faculty coaches other faculty and learning style dimensions.

- Academic performance of team members. The objective is to achieve a balance in academic performance.
- Individual Project Preference

Obviously, this selection process is not perfect but it resembles reasonably well how teams in industry are formed. Any team can work and work well if all the team members will be active and willing participants.

Individual expectations

Overview

Although the work on a project will be largely graded on a team effort, the work of a team is generally accomplished by team members doing individual work that is coordinated within the team. Thus, if a team wants to do well, individuals will have to work hard. Following are some of the expectations of an individual team member.

1. Team members must be reliable. This means that they regularly attend team meetings, contribute in a positive way, accept and complete assignments.
2. Team members will support other team members. This means that an individual will not be overly focused on his / her assignment and not help other team members as necessary. Of course, individual assignments cannot be neglected with the excuse of helping fellow team members.
3. IPPD engineers will support other IPPD engineers as necessary. Of course this requires balance as one's own team project cannot be neglected. While it is important that each team be successful, it is equally important that the program as a whole be successful.

Design notebook

Individual team members will always keep a complete design notebook. The notebook is a major deliverable and will be turned in with the final documentation of the project. *Please see the Design Notebook Specifications PDF in the "How To" folder in Canvas.*

Team member reports

Accountability: Each team member is held accountable for the assignments he/she has been given. One way of creating this accountability and also to keep the total team abreast of the project status is to have individual team members report about their assignment in the team meeting.

The individual team member report is usually accomplished at the beginning of team meetings. Team members report the number of hours spent on their assignments during the week and the progress made. This allows all team members to see if their effort and accomplishment measure up to the group. In addition, this allows team members to get help from their team in case they run into problems.

Work records: To ensure that team effort is recorded accurately, each member should keep a weekly work record. This information should be kept in the Design Notebook. Individual weekly

work records will be summarized on the team's weekly record and given to the faculty coaches weekly.

Brevity: The team member's oral report should be kept short.

Individual lecture/workshop expectations

The lecture/workshop content is closely coordinated with the progress of the industrial project. The project should give each new engineer a chance to apply the methods taught in weekly lectures and workshops. The primary goal of IPPD is the education of new engineers. The project should not be allowed to override the educational aspect.

Performance: To help meet the educational objectives a record will be kept of the performance of each new engineer. This performance will include attendance, assignments, contribution to discussions, and individual preparation.

Attendance: Attendance is mandatory at all lectures and workshops for new engineers and coaches.

Reading assignments: On the first day of the IPPD program new engineers will receive a weekly schedule overview. This will explain the lecture topics, the required reading, the workshop activities and the deliverables that are due each week. You can also find a weekly schedule overview of the class in Canvas. New engineers are expected to read the required material before the lectures.

Evaluation Policy

Guidelines

The IPPD program is an engineering education program and not just an engineering program. Evaluation of new engineers therefore is done differently than would be typical of industry. To keep an appropriate focus on the educational aspects of the experience, new engineers will be evaluated on both their performance in the lectures and workshops and their performance on the project.

For detailed information on grading, please see your IPPD syllabus, a copy of which can be found in Canvas

Classroom Labs

For detailed information on the classroom labs, please see the IPPD Professional Manual located in Canvas.

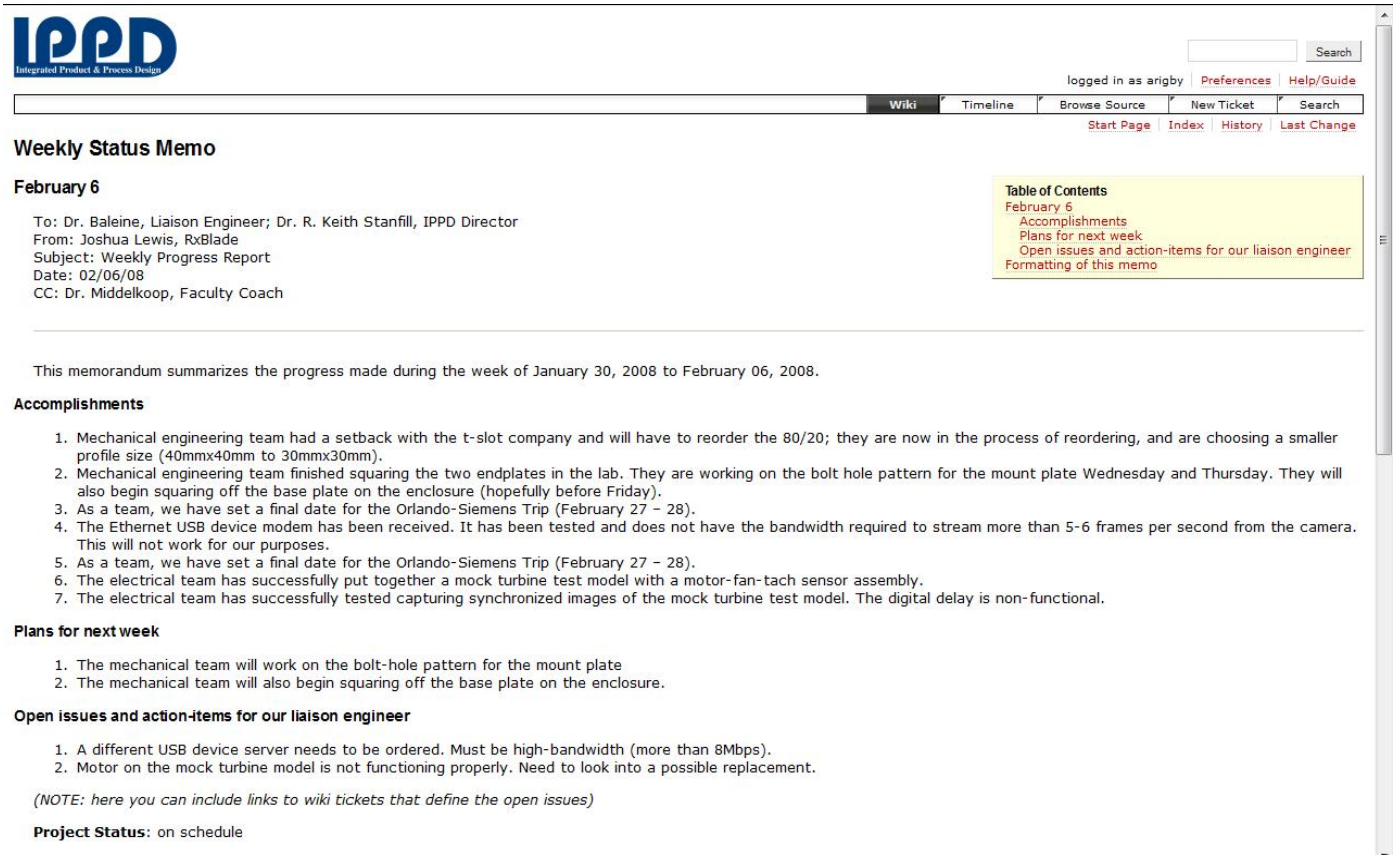
Financial Policies and Procedures

For information on Materials & Supplies requests and Travel requests, please see the IPPD Professional Manual located in Canvas.

Appendix: Weekly Status Memo

Back to Course Deliverables. Back to Weekly Team Meetings.

- The **Weekly Status Memo**, is a concise summary of the past week's accomplishments. It includes completed tasks, agreements reached, key issues, requested help. Note that this is intended to be a *progress* update -- not a *static* update (e.g. no progress to show other than "held two meetings"). The memo is to be addressed to the IPPD Director and your liaison engineer(s). The memo must be uploaded to the SVN minutes folder by the day of class each week. **Note: if you have designed your wiki to be linked to your SVN, please make sure there is also a wiki link to the file as well.*
- Although you may upload a Word document, it is preferred that you format the memo using the **WikiFormatting**. Please refer to the sample memo below. You can create a PDF of your memo from the wiki by using the WikiToPdf function located at the bottom of each trac wiki page. You may e-mail the memo to your liaison as a PDF or simply cut and paste the formatted text into your e-mail message body.



IPPD Integrated Product & Process Design

Search

logged in as arigby | Preferences | Help/Guide

Wiki | Timeline | Browse Source | New Ticket | Search

Start Page | Index | History | Last Change

Weekly Status Memo

February 6

Table of Contents
- February 6
  - Accomplishments
  - Plans for next week
  - Open issues and action-items for our liaison engineer
- Formatting of this memo

To: Dr. Baleine, Liaison Engineer; Dr. R. Keith Stanfill, IPPD Director
From: Joshua Lewis, RxBlade
Subject: Weekly Progress Report
Date: 02/06/08
CC: Dr. Middelkoop, Faculty Coach

This memorandum summarizes the progress made during the week of January 30, 2008 to February 06, 2008.

Accomplishments

1. Mechanical engineering team had a setback with the t-slot company and will have to reorder the 80/20; they are now in the process of reordering, and are choosing a smaller profile size (40mmx40mm to 30mmx30mm).
2. Mechanical engineering team finished squaring the two endplates in the lab. They are working on the bolt hole pattern for the mount plate Wednesday and Thursday. They will also begin squaring off the base plate on the enclosure (hopefully before Friday).
3. As a team, we have set a final date for the Orlando-Siemens Trip (February 27 – 28).
4. The Ethernet USB device modem has been received. It has been tested and does not have the bandwidth required to stream more than 5-6 frames per second from the camera. This will not work for our purposes.
5. As a team, we have set a final date for the Orlando-Siemens Trip (February 27 – 28).
6. The electrical team has successfully put together a mock turbine test model with a motor-fan-tach sensor assembly.
7. The electrical team has successfully tested capturing synchronized images of the mock turbine test model. The digital delay is non-functional.

Plans for next week

1. The mechanical team will work on the bolt-hole pattern for the mount plate
2. The mechanical team will also begin squaring off the base plate on the enclosure.

Open issues and action-items for our liaison engineer

1. A different USB device server needs to be ordered. Must be high-bandwidth (more than 8Mbps).
2. Motor on the mock turbine model is not functioning properly. Need to look into a possible replacement.

(NOTE: here you can include links to wiki tickets that define the open issues)

Project Status: on schedule

Appendix: Project Roadmap

The Project Roadmap: A New IPPD Deliverable to Add Flexibility to the Structured Development Process

Go back to **Final Report Process Assessment** section.

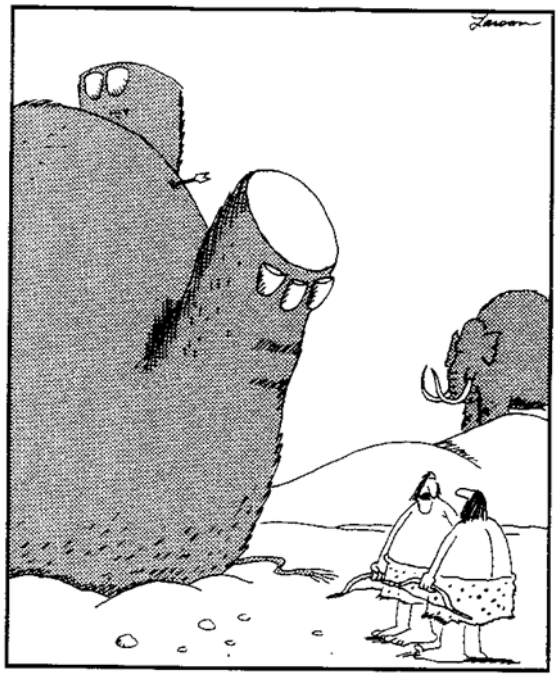
Background and motivation

The IPPD project portfolio undertaken each year has continued to grow in diversity. The project spectrum includes electronics, machines and components, software, and processes (chemical, people, manufacturing). Other dimensions of project diversity include maturity of the underlying technology and striking a balance between application-oriented and research-oriented projects.

Increasing software content in the projects drove the development of special, undocumented work-arounds to meet final project delivery needs. In the year 2000 program, the top-down structured hardware development process was complemented with a structured software development process. While this software development process has in general improved the delivery of software project results, it is not effective for all software projects. In addition, the structured hardware development process is not one-size fits-all.

How come if the defined processes do not apply to all projects, we are still consistently successful in meeting customer expectations? I think the main reason is that we have talented, experienced faculty that are adept at modifying the IPPD process as needed to get the job done. Unfortunately, we are not capturing this tribal knowledge for future projects. So, our documented process stays the same, yet the process we really use changes every year. Worse yet, each coach may invent their own new process every year.

As we develop best practices and effective strategies, we should write them down for ourselves and future project teams. Like medicine and law, engineering is a life-long learning profession. New techniques should be documented and reused.



"We should write that spot down."

Proposed deliverable: the project roadmap

The project roadmap is a statement of process. The roadmap will list the IPPD deliverables the team will be completing and an overlay of any special sponsor deliverables or review requirements. The team will need to understand the IPPD deliverables early and work with the coach and sponsor to ascertain which deliverables are applicable and the timing. The project roadmap deliverable will be due within a week of the first sponsor visit. This early process-oriented effort will help the team better understand how the project will be tackled and provide early visibility into the detailed project plan.

A partial list of things that may be included in the project roadmap:

- time for feasibility analysis
- additional prototypes
- background research phase
- a hybrid deliverable process utilizing both hardware and software deliverables
- others

A template document with "IPPD Classic" will be eventually developed the teams to use as a starting point. Teams may use any documentation tool to create the roadmap. Microsoft Project (make a high-level Gantt chart), Word, Excel, or PowerPoint could all be used. Keep it simple. If Microsoft Project is utilized, then it will be easy to transform the roadmap into a detailed project plan.

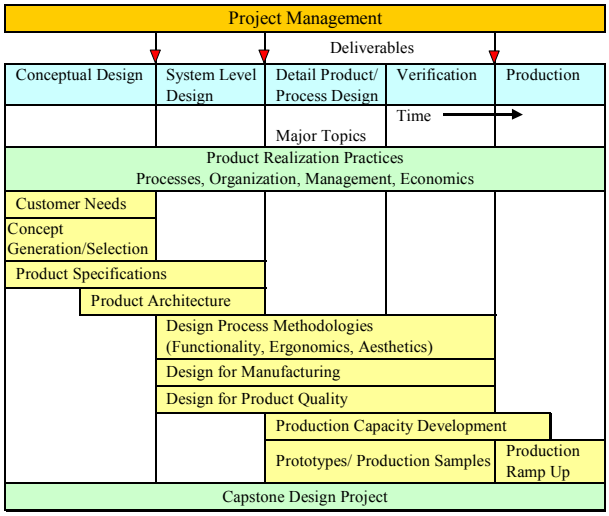
The roadmap will be submitted to the IPPD Director. Eventually a process review committee composed of at least 3 current IPPD coaches will be established to approve or recommend modifications to the roadmap. The committee will not be formalized until the 2005-2006 program year.

New section for the final report: process assessment

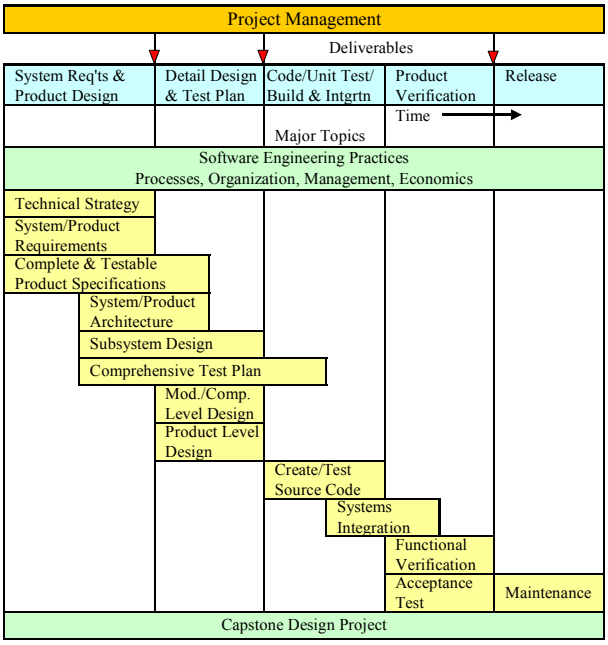
The final report will include an assessment of the process followed by the project team. This assessment will illustrate the actual roadmap followed, detail what worked and did not work, and will provide

recommendations for future projects. The project roadmap will be made available for future project teams.

Conceptual roadmap of IPPD development processes:



IPPD Development Process (hardware orientation)



IPPD Development Process (software orientation)

Appendix: Project Video Requirements

(See **Prototype Demonstration Video** section.)

Thoughts

1. keep it simple
2. keep it short
3. videos to be used within PowerPoint presentations should be less than one minute (20 seconds is optimal)
4. standalone videos should be no more than 2 minutes
5. training videos may longer, but typically would include interactive elements
6. ensure the video format and codec is compatible with PowerPoint if you plan to incorporate it
7. consider how you might incorporate the project elevator speech
8. work in the team logo and acknowledge the team members, coach, liaison, and sponsor
9. create an outline
10. develop **story boards** for the scenes you plan to show; PowerPoint is a good tool for this, but hand sketches are fine
    - Resources for story boards
      - http://multimedia.journalism.berkeley.edu/tutorials/starttofinish/storyboarding/
      - http://accad.osu.edu/womenandtech/Storyboard%20Resource/
      - http://en.wikipedia.org/wiki/Storyboard
      - http://www.sotherden.com/video101/storyboard.htm
11. create a script for any narration or interaction
12. consider using voice-overs to add narration to the video during the edit process
13. 480p is fine for most work; don't exceed 720p
14. 30 frames per second (fps) looks the best
15. secure the video equipment ahead of time
16. use a tripod

Video capture

IPPD has a Canon digital SLR that can capture still photos and video up to 1080p (15 fps). The camera can be configured to shoot in 16:9 aspect ratio and in 480p and 720p at 30 fps. Make arrangements for checking out the camera through the **IPPD inventory system**. You may of course also use your own equipment. Note that the lens quality makes a huge difference. The SLR will provide a far better image than the typical camcorder.

Video editing tools

If you use a camcorder or digital camera to capture the video, then you may wish to use the IPPD iMac computers to edit the video. iMovie works fine for most applications and can handle a variety of scene transitions, incorporation of music, and photos, plus titles and captions. Output will be in Quicktime format, but you can convert this format to others easily using tools such as **Free Video Converter.** For more complicated video editing, the iMacs have Final Cut Express (FCE). Note that FCE has a bit of a learning curve.

Software tutorial tools

An effective way to capture software demos is to use Adobe Flash. Wink is a freeware tool to capture screen movie demonstrations. Wink is available for download at **http://www.debugmode.com/wink/**. Wink runs on windows and x86 Linux distributions, but the Flash output is viewable on any platform that supports Shockwave Flash formats. Wink allows screen shots to be annotated with text captions and navigation buttons. Sound files can be mixed in and voice can be over-dubbed (sound is sampled at 11 kHz, so it will be a little distorted). Audacity is a freeware sound editing tool that can be used to record voice-over for the wink demo segments. Audacity is available at **http://audacity.sourceforge.net/**.

A decent microphone is essential if you wish to do voice-overs. Finding a quiet place to record will give you the best shot at a clean recording. Combination headphone-microphones may be useful, but be wary of pop and hiss from your normal speaking voice. You may need to position the mic up even with the end of your nose and adjust the recording levels to max out at 0 db. USB mics from Guitar Hero or Rock Band games will probably work fine, though you may need to add a wind screen (foam) to the mic and place a pop filter (stretch nylon hose over an embroidery hoop) between you and the mic.

There may be other screen capture tools out there, so please search away. There are likely several OS X tools that will work fine on the Mac.