TEMPERATURE CONTROL SYSTEM

ENPM642

Presented to:

Professor Mark Austin

Students:

Josemaria Dolores

Francisco Gallo

Introduction

- This project is a continuation of the Temperature Control System (TCS) done in previous semester (Course: ENPM641).
- This is a system designed to control the temperature in a closed space based on certain user defined parameters.

Results from last semester

- All the documents and deliverables from previous semester are available at http://www.wam.umd.edu/~fgallo/ENPM641/Index.htm
- Last semester the final products obtained were:
  - Initial requirements, goals, scenarios.
  - High level models of system structure and behavior.
  - Mapping of high level behavior onto high level system structure.
  - Formulation of some optimization alternatives.

Goals for this semester

1. Define quantitative requirements for the project at a system level and a component level.
2. Ensure that the components selected can communicate among each other by evaluating its input/output characteristics and constrains.
3. Perform optimization and tradeoff analysis so the best combination of components can be selected for a given set of decision criteria.
4. Perform appropriate traceability of requirements to ensure the components and configuration selected are able to fulfill the expected performance and behavior parameters.

Restrictions and assumptions

- In ENPM641 the TCS was conceived as a generic system that could be installed into any room or space. In ENPM642 the system will be instantiated and optimized for a specific space. This space is a warehouse with the following dimensions: Height= 8 feet. Width=100 feet. Length= 100 feet. The warehouse will be considered as a space poorly insulated.
- The project focuses on the following components:
  - **Thermometer**
  - **Controlling device** → for example, a PID or other.
  - **Programmable parameters controlling device (PPCD)** → for example a PC where the user can program parameters such as desired temperature.
  - **Fan**
  - **Heater**
  - **Chiller**

Restrictions and assumptions

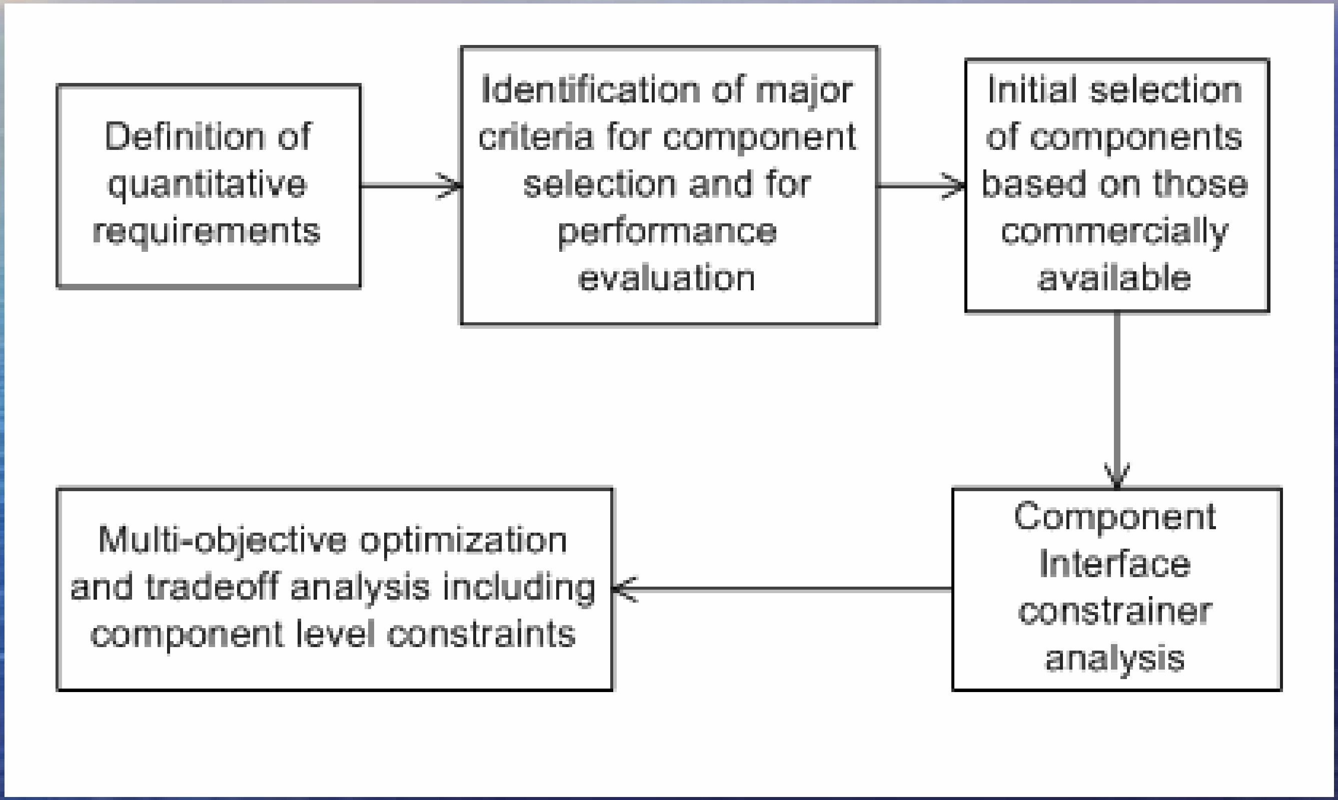
- Some simulation tools available on the Internet have been used to generate estimates (e.g. BTUs and similar computations).

- These simulation tools make certain assumptions based on the expected average temperature and general weather conditions of the place where the system to control de temperature is going to be installed.

- In order to provide a context for these suppositions, all the calculations done for this project were done assuming that the warehouse is located in Virginia.

- The simulators used to determine the BTUs are at:
  - http://hearth.com/calc/btucalc.html
  - http://www.a-wall.com/ACcalc.php

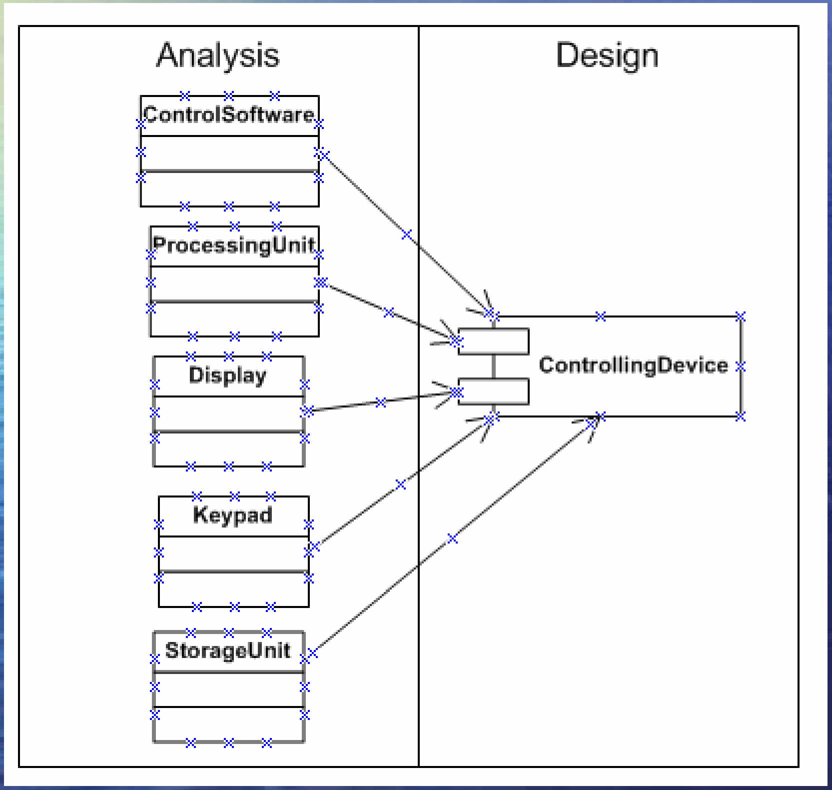
General strategy for the project

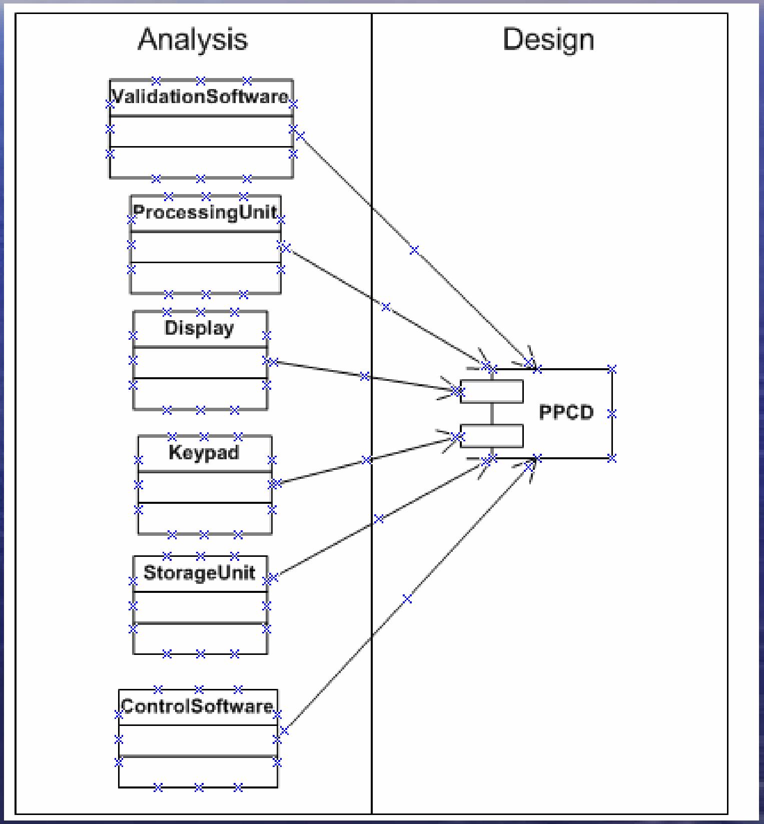


Classes traceability

- ENPM641 was focused on the Analysis workflow. This workflow is abstract.
- In ENPM642, the Design workflow was more important.
- Analysis classes were mapped into Design classes (components).

Classes traceability

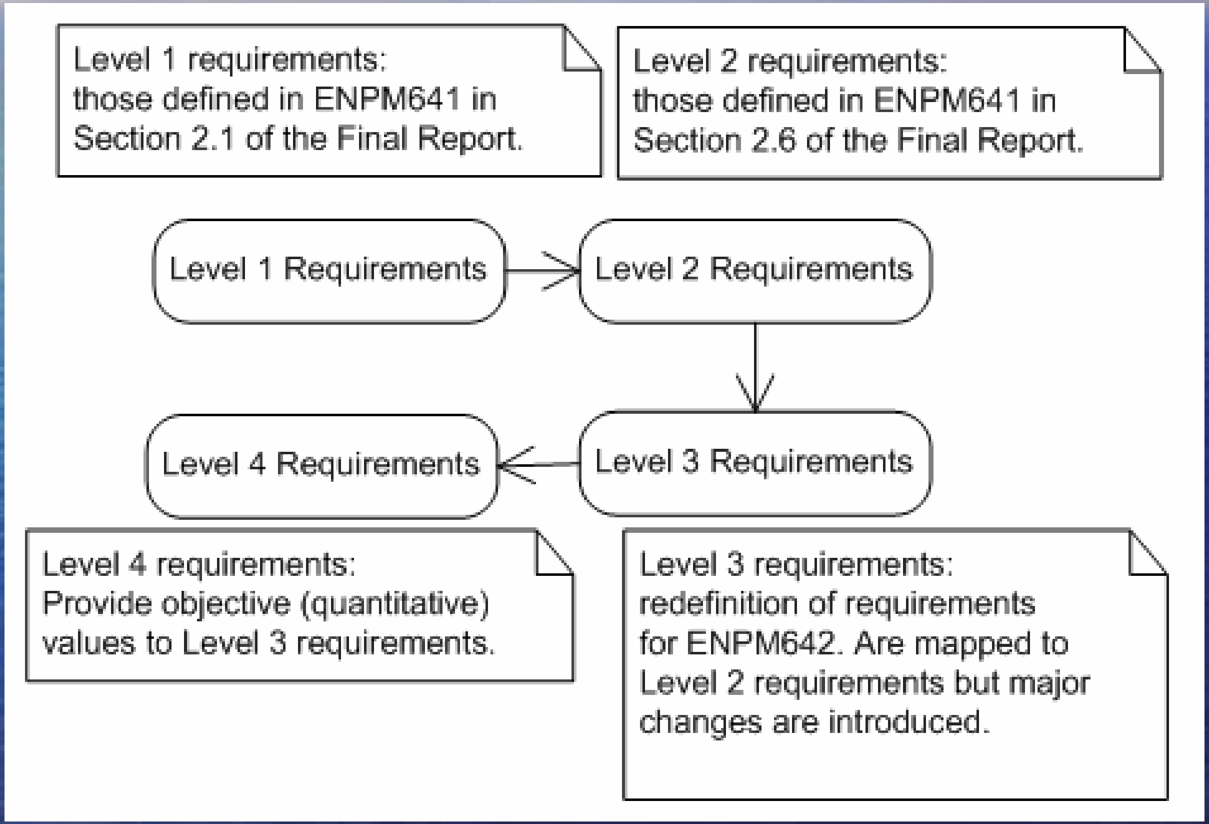




Requirements traceability

- Several levels of requirements were defined:
  - **Level 1:** Corresponds to requirements defined in ENPM641.
  - **Level 2:** Corresponds to refined requirements defined in ENPM641.
  - **Level 3:** Corresponds to the re-definition of the requirements based on the assumptions and restrictions from Section 2 in this document.
  - **Level 4:** Corresponds to the requirements-specifications (req-specs) where objective (quantitative) parameters have been defined.

Requirements traceability



Consolidation of requirements

- Each of the quantitative requirements (Level4) were mapped into each of the components of the system.
- The criteria for component selection were defined:
  - Initial acquisition cost.
  - Durability of system.
  - Reliability of system.
- Criteria for Performance Evaluation:
  - Efficiency of energy usage of the system (i.e. minimum waste, maximum power transfer to actual load).
  - Accuracy of components.
  - Speed of system to reach desired conditions.
  - Power output (i.e. capability of the system to perform given the magnitude of the element to be controlled)→ e.g. BTUs of heating unit.

Component Interface Constrainer Application

You may be asking yourself… What is this guy going to talk about?

- 1) Why did we choose to do constrain components by interface specification?
- 2) How did we write a program to do it?
- 3) What’s the concept behind this program? Can I see some screen shots?
- 4) Can I please see a demonstration? SURE!

Problem Statement

Past student projects have optimized components from a component library without taking into account the the interface specifications between the components. The Component Interface Constrainer Program addresses this issue by allowing a user to constrain a component library based on interface limitations.

Program Attributes

- Java Standard Edition version 1.4.2
- JDOM version 1.0
- XML Database represented as XML file
- XSL Transformations
- Log4j logging
- Properties file configuration

JDOM

- JDOM is a Java representation of an XML document.
- It is an open source interface which allows Java developers to manipulate XML based components in a programmatic way.

Why XML?

- XML is the new standard for interoperability between databases.
- It’s strength lies in embedding metadata with data in a hierarchical structure.
- This allows processing components to be loosely coupled with the data being processed.
- A well defined schema allows this process to be extended to any kind of component library.

What about XSL?

- XSL stands for <u>X</u>ML <u>S</u>tylesheet <u>L</u>anguage.
- XSL allows a developer to separate the content of XML data from the layout of that data.
- XSL can be used to format to GUI or a back end component.
- Since XSL is an instance of XML, it benefits from all XML has to offer.

Log4j

- Logging is crucial for system feedback.
- Log4J enables a developer to output statements as a program runs.
- It’s main advantage over the standard output print statement is that it allows for runtime configurability, filtering, well as multicasting output to different sources.

Properties File

- By referencing a properties file a system becomes configurable at runtime and more variables can be set by the user as the system initializes.
- The Component Interface Constrainer Program can set which stylesheet it will use at run time to determine the output.

Algorithm

- For n number of constraints on i components, where i can be: Controller, Heater, Chiller, Fan, Thermometer, or PPCD
- Choose that type of component from the component library and keep it only if it fulfills the current constraint.
- Output all the “passing” components to an html file so the user can easily browse the feasible set.

Specification Description

- The key to the process of constraining components based on interface specifications is in ***describing*** the input and output specifications in a database which can be processed.
- The extensibility of the database is a direct cause of how well the system can be improved and expanded if later functionality is desired.

Java Code Picture

```
/**
 * For each constraint, go through each component of that constraint's type and see if that component fulfills the constraint.
 *
 * @param componentDatabaseAsHashMap The component databas as HashMap divided by component type
 * @param currQueryComponentType The component type being constrained
 * @param currQueryConstraintType The constraint type of the interface spec: either input or output
 * @param currQueryConnectingComponent The component being communicated with
 * @param currQuerySpecName The name of the specification
 * @param currQuerySpecValue The value of the specification
 * @param currQueryOperator Whether the value should be =, <, or >
 * @return The a new component database as HashMap, but without the components that did not meet the constraint
 */
private static HashMap constrainComponentSet(HashMap componentDatabaseAsHashMap, String currQueryComponentType,
    String currQueryConstraintType, String currQueryConnectingComponent, String currQuerySpecName, String currQuerySpecValue, String currQueryOperator) {

  //Create a new output element which will replace this component container in the hash map.
  Element containerComponent = new Element(currQueryComponentType, xmlDatabaseNamespace);

  //Pull the appropriate components (based on type) out of the HashMap.
  Element constrainedComponents = (Element) ((Element) componentDatabaseAsHashMap.get(currQueryComponentType)).clone();
  List componentList = constrainedComponents.getChildren();
  Iterator componentIter = componentList.iterator();
  while(componentIter.hasNext()) {
    Element currentElem = (Element) componentIter.next();

    //String currElemComponentName = currentElem.getChildText("ComponentName", xmlDatabaseNamespace);
    logger.info("Evaluating the component: " + currentElem.getChildText(TCSConstants.COMPONENT_NAME_ELEM, xmlDatabaseNamespace));

    //Get the specification for each connection this component can have, either all the input components or all the output components.
    List connectedComponentList = null;
    if(currQueryConstraintType.equalsIgnoreCase(TCSConstants.INPUT_ELEM)){
      connectedComponentList = currentElem.getChild(TCSConstants.COMPONENT_INPUT_SPEC_ELEM,xmlDatabaseNamespace).getChildren(TCSConstants.CONNECTED_COMPONENT_ELEM, xmlDatabaseNames
      logger.info("Looking at the input specs.");
      List possibleConnectedComponentsList = currentElem.getChild(TCSConstants.COMPONENT_INPUT_SPEC_ELEM,xmlDatabaseNamespace).getChild(TCSConstants.POSS_COMPONENT_TYPES_ELEM, xmlDa
      Iterator possibleConnectedComponentsIter = possibleConnectedComponentsList.iterator();
      StringBuffer possibleConnectedComponents = new StringBuffer();
      int possConnCompCounter = 0;
      while(possibleConnectedComponentsIter.hasNext()) {
        possConnCompCounter++;

        //If this is the second component or more to be added to the list, add a comma to the list first.
        if(possConnCompCounter > 1){
          possibleConnectedComponents.append(", ");
        }
        possibleConnectedComponents.append(((Element) possibleConnectedComponentsIter.next()).getText());
      }
      logger.debug("This component can recieve input from the following components: " + possibleConnectedComponents.toString() + ".");
```

Log4J Picture

```
INFO - ____________________NEW CONSTRAINT______________________
INFO - Constraint number 1: The Controller component's Input Signal must be = 7V when it is connected to the PPCD.
INFO - Evaluating the component: CALController 1200
INFO - Looking at the input specs.
INFO - Looking at the spec when this Controller is connected to a PPCD.
INFO - Looking at the Controller's Input Signal specification.
INFO - The max value for this component's Input Signal is: 5.0.
INFO - The min value for this component's Input Signal is: -5.0.
INFO - Discard this component.
INFO - Evaluating the component: CALController 3300
INFO - Looking at the input specs.
INFO - Looking at the spec when this Controller is connected to a PPCD.
INFO - Looking at the Controller's Input Signal specification.
INFO - The max value for this component's Input Signal is: 25.0.
INFO - The min value for this component's Input Signal is: -25.0.
INFO - Keep this component.
INFO - Evaluating the component: CALController 9300
INFO - Looking at the input specs.
INFO - Looking at the spec when this Controller is connected to a PPCD.
INFO - Looking at the Controller's Input Signal specification.
INFO - The max value for this component's Input Signal is: 25.0.
INFO - The min value for this component's Input Signal is: -25.0.
INFO - Keep this component.
INFO - Evaluating the component: CALController 9500
INFO - Looking at the input specs.
INFO - Looking at the spec when this Controller is connected to a PPCD.
INFO - Looking at the Controller's Input Signal specification.
INFO - The max value for this component's Input Signal is: 25.0.
INFO - The min value for this component's Input Signal is: -25.0.
INFO - Keep this component.
INFO - ____________________NEW CONSTRAINT______________________
INFO - Constraint number 2: The Heater component's Input Signal must be = 7V when it is connected to the Controller
INFO - Evaluating the component: Sterling 250 TF
INFO - Looking at the input specs.
INFO - Looking at the spec when this Heater is connected to a Controller.
INFO - Looking at the Heater's Input Signal specification.
INFO - The max value for this component's Input Signal is: 25.0.
INFO - The min value for this component's Input Signal is: -25.0.
INFO - Keep this component.
INFO - Evaluating the component: Sterling 300 TF
INFO - Looking at the input specs.
INFO - Looking at the spec when this Heater is connected to a Controller.
INFO - Looking at the Heater's Input Signal specification.
INFO - The max value for this component's Input Signal is: 10.0.
INFO - The min value for this component's Input Signal is: -10.0.
INFO - Keep this component.
INFO - Evaluating the component: Sterling 450 TF
INFO - Looking at the input specs.
```

XML Database Picture


```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<tcs:ComponentLibrary xmlns:tcs="http://www.wam.umd.edu/~fgallo/ENPM642/Project/TCSNamespace">
	<tcs:Component>
		<tcs:ComponentType>Controller</tcs:ComponentType>
		<tcs:ComponentName>CALController 1200</tcs:ComponentName>
		<tcs:ComponentGeneralCharacteristics>
			<tcs:Characteristic>
				<tcs:CharacteristicName tcs:abbreviation="null">Number of Displays</tcs:CharacteristicName>
				<tcs:CharacteristicUnit tcs:abbreviation="null">null</tcs:CharacteristicUnit>
				<tcs:CharacteristicValue>1</tcs:CharacteristicValue>
				<tcs:CharacteristicDescriptionText>The Display has 4 LED digits which are 10 mm. in height.</tcs:CharacteristicDescriptionText>
			</tcs:Characteristic>
			<tcs:Characteristic>
				<tcs:CharacteristicName tcs:abbreviation="null">Input Buttons</tcs:CharacteristicName>
				<tcs:CharacteristicUnit tcs:abbreviation="null">null</tcs:CharacteristicUnit>
				<tcs:CharacteristicValue>3</tcs:CharacteristicValue>
				<tcs:CharacteristicDescriptionText>The Display has 3 elastrometric input buttons.</tcs:CharacteristicDescriptionText>
			</tcs:Characteristic>
			<tcs:Characteristic>
				<tcs:CharacteristicName tcs:abbreviation="null">Number of Output Ports</tcs:CharacteristicName>
				<tcs:CharacteristicUnit tcs:abbreviation="null">null</tcs:CharacteristicUnit>
				<tcs:CharacteristicValue>3</tcs:CharacteristicValue>
				<tcs:CharacteristicDescriptionText>This product has 3 output ports.</tcs:CharacteristicDescriptionText>
			</tcs:Characteristic>
			<tcs:Characteristic>
				<tcs:CharacteristicName tcs:abbreviation="null">Warranty</tcs:CharacteristicName>
				<tcs:CharacteristicUnit tcs:abbreviation="yrs">Years</tcs:CharacteristicUnit>
				<tcs:CharacteristicValue>1</tcs:CharacteristicValue>
				<tcs:CharacteristicDescriptionText>The warranty for this product is 1 year.</tcs:CharacteristicDescriptionText>
			</tcs:Characteristic>
			<tcs:ComponentGeneralDescriptionText>CALController 1200 Controller is our worst controller ever!!!</tcs:ComponentGeneralDescriptionText>
		</tcs:ComponentGeneralCharacteristics>
		<tcs:ComponentOptimizableCharacteristics>
			<tcs:Characteristic>
				<tcs:CharacteristicName tcs:abbreviation="null">Acquisition Cost</tcs:CharacteristicName>
				<tcs:CharacteristicUnit tcs:abbreviation="USD">United States Dollars</tcs:CharacteristicUnit>
				<tcs:CharacteristicValue>600000.00</tcs:CharacteristicValue>
				<tcs:CharacteristicDescriptionText>The cost may vary on market prices.</tcs:CharacteristicDescriptionText>
			</tcs:Characteristic>
			<tcs:Characteristic>
				<tcs:CharacteristicName tcs:abbreviation="null">Reliability</tcs:CharacteristicName>
				<tcs:CharacteristicUnit tcs:abbreviation="%">Percent</tcs:CharacteristicUnit>
				<tcs:CharacteristicValue>75.00</tcs:CharacteristicValue>
				<tcs:CharacteristicDescriptionText>Reliability is measured by internal testing departments.</tcs:CharacteristicDescriptionText>
			</tcs:Characteristic>
			<tcs:Characteristic>
				<tcs:CharacteristicName tcs:abbreviation="null">Durability</tcs:CharacteristicName>
				<tcs:CharacteristicUnit tcs:abbreviation="hrs">Hours</tcs:CharacteristicUnit>
				<tcs:CharacteristicValue>120000.00</tcs:CharacteristicValue>
				<tcs:CharacteristicDescriptionText>Durability is a market specification provided by the Vendor.</tcs:CharacteristicDescriptionText>
			</tcs:Characteristic>
		</tcs:ComponentOptimizableCharacteristics>
		<tcs:ComponentInputSpecification>
			<tcs:PossibleComponentTypes>
				<tcs:ComponentType>Thermometer</tcs:ComponentType>
				<tcs:ComponentType>PPCD</tcs:ComponentType>
			</tcs:PossibleComponentTypes>
			<tcs:ConnectedToComponent>
				<tcs:ComponentType>PPCD</tcs:ComponentType>
				<tcs:Spec>
					<tcs:Range>N</tcs:Range>
					<tcs:SpecName tcs:abbreviation="null">Input Signal Type</tcs:SpecName>
					<tcs:SpecUnit tcs:abbreviation="null">null</tcs:SpecUnit>
					<tcs:SpecValue>Analog</tcs:SpecValue>
					<tcs:SpecDescriptionText>null</tcs:SpecDescriptionText>
				</tcs:Spec>
				<tcs:Spec>
					<tcs:Range>N</tcs:Range>
					<tcs:SpecName tcs:abbreviation="null">Link Type</tcs:SpecName>
					<tcs:SpecUnit tcs:abbreviation="null">null</tcs:SpecUnit>
					<tcs:SpecValue>Serial</tcs:SpecValue>
```

XSL Picture

```
<!-- Match the root node. -->
<xsl:template match="/">
        <xsl:element name="html">
                <xsl:element name="body">

                        <!-- Create a header in the output document with the same text as the title. -->
                        <xsl:element name="h1"><xsl:value-of select="'Feasible Component Optimizable Specifications'" /></xsl:element>
                        <xsl:apply-templates select="tcs:ComponentLibrary" />
                </xsl:element>
        </xsl:element>
</xsl:template>

<!-- Match the components in order of their type so the output document is sorted in order of type. -->
<xsl:template match="tcs:ComponentLibrary">
        <xsl:apply-templates select="tcs:Component[tcs:ComponentType='Controller']" />
        <xsl:apply-templates select="tcs:Component[tcs:ComponentType='Heater']" />
        <xsl:apply-templates select="tcs:Component[tcs:ComponentType='Chiller']" />
        <xsl:apply-templates select="tcs:Component[tcs:ComponentType='Fan']" />
        <xsl:apply-templates select="tcs:Component[tcs:ComponentType='Thermometer']" />
        <xsl:apply-templates select="tcs:Component[tcs:ComponentType='PPCD']" />
</xsl:template>

<!-- Create a table for each component with a Caption holding the type and name of the component. -->
<xsl:template match="tcs:Component">
        <xsl:element name="table">
                <xsl:attribute name="border">1</xsl:attribute>
                <xsl:element name="caption">
                        <xsl:element name="b">
                                <xsl:value-of select="tcs:ComponentType" /><xsl:text>: </xsl:text><xsl:value-of select="tcs:ComponentName" />
                        </xsl:element>
                </xsl:element>
                <xsl:element name="tr">
                        <xsl:element name="th"><xsl:text>Attribute</xsl:text></xsl:element>
                        <xsl:element name="th"><xsl:text>Value</xsl:text></xsl:element>
                        <xsl:element name="th"><xsl:text>Text Description</xsl:text></xsl:element>
                </xsl:element>
                <xsl:apply-templates select="tcs:ComponentOptimizableCharacteristics" />
        </xsl:element>

        <!-- Separate each table with a line. The <br /> tag does not seem to work. -->
        <xsl:element name="p" />
</xsl:template>

<!-- Populate the table with the Characteristics outlined. -->
<xsl:template match="tcs:ComponentOptimizableCharacteristics">
        <xsl:apply-templates select="tcs:Characteristic" />
</xsl:template>

<!-- Use the Characteristic Name, Unit Value, and Text Description to populate the table. -->
<xsl:template match="tcs:Characteristic">
        <xsl:element name="tr">
                <xsl:element name="th">
                        <xsl:value-of select="tcs:CharacteristicName" />
                        <xsl:if test="tcs:CharacteristicUnit/@tcs:abbreviation!='null'">
                                <xsl:text>(</xsl:text>
                                <xsl:value-of select="tcs:CharacteristicUnit/@tcs:abbreviation" />
                                <xsl:text>)</xsl:text>
                        </xsl:if>
                        <xsl:text>:</xsl:text>
                </xsl:element>
                <xsl:element name="td"><xsl:value-of select="tcs:CharacteristicValue" /></xsl:element>
                <xsl:choose>
                        <xsl:when test="tcs:CharacteristicDescriptionText!='null'">
                                <xsl:element name="td"><xsl:value-of select="tcs:CharacteristicDescriptionText" /></xsl:element>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:element name="td"><xsl:text>N/A</xsl:text></xsl:element>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:element>
</xsl:template>
```

XML + XSL = HTML

Feasible Component Optimizable Specifications

Controller: CALController 1200

Attribute	Value	Text Description
Acquisition Cost(USD):	600000.00	The cost may vary on market prices.
Reliability(%):	75.00	Reliability is measured by internal testing departments.
Durability(hrs):	120000.00	Durability is a market specification provided by the Vendor.

Controller: CALController 3300

Attribute	Value	Text Description
Acquisition Cost(USD):	23000.00	The cost may vary on market prices.
Reliability(%):	98.99	Reliability is measured by internal testing departments.
Durability(hrs):	450000.00	Durability is a market specification provided by the Vendor.

Controller: CALController 9300

Attribute	Value	Text Description
Acquisition Cost(USD):	25300.00	The cost may vary on market prices.
Reliability(%):	99.15	Reliability is measured by internal testing departments.
Durability(hrs):	800000.00	Durability is a market specification provided by the Vendor.

Controller: CALController 9500

Attribute	Value	Text Description
Acquisition Cost(USD):	23850.00	The cost may vary on market prices.
Reliability(%):	99.85	Reliability is measured by internal testing departments.
Durability(hrs):	510000.00	Durability is a market specification provided by the Vendor.

Heater: Sterling 250 TF

Attribute	Value	Text Description
Power Output(BTU):	310000.00	This is the main index of measurement which differentiates between heaters in industry.
Free Air Delivery(CFM):	3500	Cubic Feet per Minute is a standard used in industry.
Efficiency(%):	94.00	Efficiency Index is a market specification provided by the vendor.

OO = Easier Improvement

- This program is object oriented, meaning it could be connected to other objects and can be modularly improved.
- Possible improvements include:
  - Graphical user interface for constraint query
  - Graphical user interface for data entry
  - Schema Validation for all data entry
  - Connection to optimization component
  - Recursive component interface analysis
  - Ported to Web Application

Demo

Optimization and tradeoff analysis

- **Goal**: select the optimum combination of components based on the set of options obtained from the Component Interface Constrainer Application.
- Use of Excel and Visual Basic for Applications (VBA).

Optimization and tradeoff analysis

- Definition of design objectives:
  1. Initial acquisition cost.
  2. Durability of system.
  3. Reliability of system.
- Parameters for evaluation of performance were also included in this part of the analysis.

Component:	**Heater**					
Parameters of interest:						
Parameter	**Units**	**Value Min**	**Value Max**	**Heater1**	**Heater2**	**Heater3**
Power - Output	*BTU*	160,000.00	320,000.00	310,000.00	120,000.00	290,000.00
Airflow	*CFM*	3,000.00	4,000.00	3,500.00	3,100.00	3,750.00
Efficiency	*%*	75.00	100.00	94.00	99.00	92.00
Cost	*USD*	0.00	35,000.00	32,000.00	27,000.00	31,500.00
Durability	*Hours*	100,000.00	1,000,000.00	220,000.00	350,000.00	350,000.00
Reliability	*%*	98.00	100.00	99.50	99.70	99.15
List validation ends here				[EndComponent]	[EndComponent]	[EndComponent]

Optimization and tradeoff analysis

- After having all components, the summary based on the design objectives and the constraints is generated.
- All the information was consolidated using dynamically linked references.

	A	B	C	D	E	F	G	H
1	Optimization General Formulation							
2								
3	Design Objectives							
4	Initial Acquisition Cost [USD]				Reliability [%]			Durability [Hours]
5	Component	Option1	Option2	Option3	Option1	Option2	Option3	Option1
6	Heater	32,000.00	27,000.00	31,500.00	99.50	99.70	99.15	220,000.00
7	Chiller	25,000.00	27,500.00	18,000.00	99.99	99.97	99.97	540,000.00
8	Fan	1,350.00	1,200.00	1,400.00	99.78	99.98	99.85	350,000.00
9	PCCD	4,500.00	4,240.00	4,700.00	99.95	99.75	99.99	500,000.00
10	ControllingDev	23,000.00	25,300.00	23,850.00	98.99	99.15	99.85	450,000.00
11	Thermometer	850.00	1,350.00	750.00	99.95	99.98	99.55	750,000.00
12	End of list of components							

Optimization and tradeoff analysis

This is the range that delimits each constraint.

This is the value for each constraint for the option shown in column G.

	A	B	C	D	E	F	G	H	I	J
14	**Constraints**									
15	**Constrai**	**Component it applie**	**Constraint**	**Units**	**Value Min**	**Value Max**	**Option # [1 to 3]**	**Value of selected option**	**Constraint Validation**	**Component Validation**
16	1	Heater	Power - Output	BTU	160,000.00	320,000.00	2	120,000.00	FAILS	HEATER FAILS
17	2	Heater	Airflow	CFM	3,000.00	4,000.00	2	3,100.00	PASSES	
18	3	Heater	Efficiency	%	75.00	100.00	2	99.00	PASSES	
19	4	Heater	Cost	USD	0.00	35,000.00	2	27,000.00	PASSES	
20	5	Heater	Durability	Hours	100,000.00	1,000,000.00	2	350,000.00	PASSES	
21	6	Heater	Reliability	%	98.00	100.00	2	99.70	PASSES	
22	7	Chiller	Power - Output	BTU	200,000.00	350,000.00	3	300,000.00	PASSES	CHILLER OK
23	8	Chiller	Airflow	CFM	3,000.00	4,000.00	3	3,900.00	PASSES	
24	9	Chiller	Efficiency	%	68.00	100.00	3	94.00	PASSES	
25	10	Chiller	Cost	USD	0.00	28,000.00	3	18,000.00	PASSES	
26	11	Chiller	Durability	Hours	100,000.00	1,000,000.00	3	550,000.00	PASSES	
27	12	C				100.00	3	99.97	PASSES	
28	13	F				4,000.00	3	3,950.00	PASSES	FAN OK
29	14	F				100.00				
30	15	F				100.00				

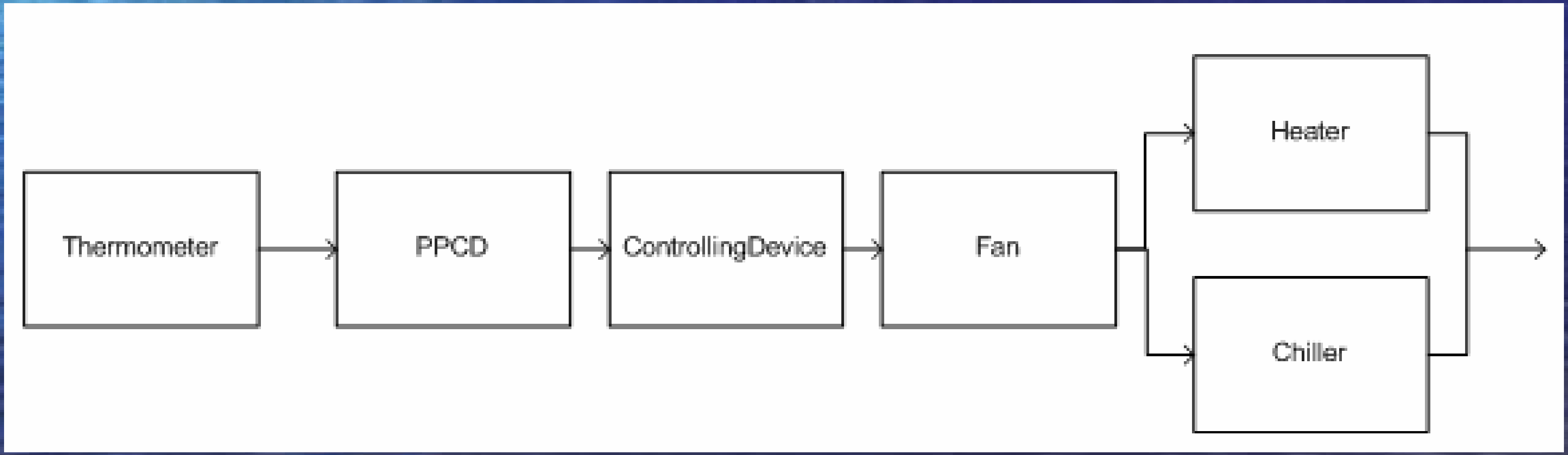
The option evaluated in column G is dynamically generated so each possible alternative is tested.

Column I evaluates each constraint for each option.

Models used for the design objectives

- Cost: sum of each individual cost
- Reliability:

  *Rel = RelTher * RelPPCD * RelContDevice * RelFan * {1-(1-RelHeater)(1-RelChiller)}*



Models used for the design objectives

- **Durability**: The system durability was calculated based on the durability of each individual component and its cost. The assumption is that if an expensive component fails, the overall durability of the system is reduced more dramatically than if a cheaper component fails and needs to be repaired or replaced.

- Durability = **{**(DurHeater x CostHeater) + (DurChiller x CostChiller) + (DurFan x CostFan) + (DurContrDevice x CostContDevice) + (DurPPCD x CostPPCD) + (DurTherm x CostTherm)**}** / TotCostofSystem

Optimization and tradeoff analysis

Excel and VBA Demo of final steps for the Optimization and Tradeoff analysis:

1. Clear the data
2. Run unconstrained routine
3. Show charts
4. Run constraints
5. Show messages of excluded components
6. Show charts with constraints
7. Show tab with the final results and weighted sums.

Questions