

## A Four-step Method for Capstone Design Teams to Gather Relevant and Well-defined Product Requirements

**Dr. Rachana Ashok Gupta, North Carolina State University**

Dr. Rachana A Gupta is currently a Teaching Associate professor and Associate Director of ECE Senior Design Program at NCSU. She teaches and mentors several senior design students on industry-sponsored projects (On average 12 / semester) to successful completion of an end product. These projects include all aspects of System Engineering: concept design, product design and design trade-offs, prototyping and testing (circuit design, PCB, mechanical fabrication, algorithm development). These projects have included Robotics Platforms, Planning, Monitoring and Control algorithms, Sensor Interface, User Interfaces, Wireless communication, Signal Processing etc. All of this involves direction and teaching teams how to use the required tools and apply engineering skills to transform a concept into a product. She also manages interdisciplinary senior design projects in collaboration with other engineering departments such as Textiles Engineering, Mechanical engineering, etc. Beyond senior design, she has also created and teaches undergraduate as well as graduate level classes in ECE (Python in Engineering, Algorithms in ECE, Practical Engineering Prototyping (PrEP)). She also has designed and taught ECE Robotics summer camp since 2012.

Dr. Gupta earned her Bachelors of Engineering in Electronics and Telecommunication Engineering from University of Pune, India and received her MS and PhD in Electrical and Computer Engineering from North Carolina State University (2010). Her Phd was to design computer vision algorithms for autonomous navigation for cars. She started her own engineering consulting company in 2010 worked on several product development projects such as automated air suspension system for vehicles, active suspension system for heavy duty off-road vehicles (currently DARPA funded), vision tracking system for race car tracks, etc. She joined NCState as Teaching Assistant Professor in 2012. Dr. Gupta's current research projects focus on sensor systems and engineering design education.

Dr. Gupta likes to tinker with new technology and work on small hobby projects in her basement lab. Her other hobbies include reading, classical dancing, and traveling.

**Mr. Greg A. Dunko, NantHealth**

Greg is the Senior Vice President of the Product and Program Management Office (PPMO) at NantHealth, where he leads strategic product planning and program business operations. Prior to joining NantHealth, he served as Global Head of Product Development at BlackBerry, leading all mobile phone hardware development. Prior to this, Greg led the Electrical and Computer Engineering senior design program at North Carolina State University – creating a new full year program with emphasis on product development and corporate sponsorship and mentoring. He has also held leadership roles at HTC and Ericsson/Sony Ericsson. Greg is an established inventor and has filed over 80 patents. He also is co-author of the eBook "A Reference Guide to the Internet of Things". Greg holds an BS Electrical Engineering and MS Electrical Engineering from West Virginia University. His graduate research focused on Biomedical Engineering.

## **A 4-step Method for Capstone Design Teams to Gather Relevant and Well-defined Product Requirements**

---

### **PROBLEM BACKGROUND**

MITRE's System Engineering Guide [1] states, "Requirements define the capabilities that a system must have (functional) or properties of that system (non-functional) that meet the users' needs to perform a specific set of tasks (within a defined scope)." One of the challenges faced in engineering education is to teach students how to think about the element of "what" in a project before they get into the aspect of "how." In design classes, as in industry, this is mainly encapsulated in the "Product Requirement" phase of System Engineering. According to many system engineering studies [8],[10],[11], design course outcomes, and in industry, learning how to develop good product requirements is a critical element of System Engineering. A comprehensive and unambiguous list of product requirements, while keeping the user in mind, sets the correct tone for the preliminary design exercises, improves the system design, informs key design decisions, reduces the design and development time, effort, budget, and reduces the probability of requiring rework in a project.

As described in [2],[3],[8] as well as in the authors' industrial experience, the process of eliciting and developing good Product Requirements is iterative, which involves discussions with stakeholders, potential customers (i.e., Serviceable Market) and with the product sponsoring agency. However, most of the engineering subjects and courses are designed to train students to think about "how" to solve a given problem instead of the "what and where." Many students go through the "requirement" exercise only during required design courses such as Freshman Design and Capstone Design in the context of a team project. Before enrolling in the Capstone design class, only approximately 10% of the students, based on a survey done by the authors, have been introduced to the skills and tools required to interact with real customers/clients to learn how to understand their requirements (*A survey including ~ 200 senior undergraduate students (~50 teams, all working on different project with different sponsoring entity) from North Carolina State University's Electrical and Computer Engineering department from last 3 years was conducted to understand the problem. Survey was done after introducing what are product requirements and the resources to go about them but before teaching the 4-step method explained here.*) Therefore, it is not uncommon for a student team to not only struggle to understand the value of good "product requirements" but also to struggle to draft a comprehensive list of product requirements to define the scope of the project. This is true especially when a student team is given an ambiguous or an open-ended problem or conflicting customer requirements / needs, occurrences of which are typical in industrial settings.

Although there are numerous resources available [1]-[5],[9] that describe how to gather and define effective product requirements, it is authors' experience that undergraduate students struggle with devising questions to gain a comprehensive understanding of the specific needs of the customer and sponsors/stakeholders. Solution to this struggle is not addressed in these System Engineering resources. Several of these resources cannot be taught in a few lecture sessions or exercised as in-class activities. Based on our survey, 80 to 90 % of students face this problem irrespective of the available resources. The novelty of the 4-step method/approach

introduced in this paper is the logical breakdown of the required steps for defining product requirements into simple questions and tasks, which can be quickly incorporated as in-class workshops, peer review exercises, case studies of previous projects and/or sponsor meetings. Moreover, the method tasks and questions can be applied to a wide variety of project types (e.g., hardware, software, feasibility study, open-ended problem). Tasks and tools help the team keep focused on the “what” of the problem at hand. A case study conducted as an in-class workshop is provided at the end of the paper as an example of the 4-step approach. The authors assume that the teams are given resources to understand the data about market/customer and existing technologies in similar products before or during the rigorous “Product Requirements” exercise.

#### 4-STEP APPROACH:

To help students think about the correct “whats” and craft a customer need relevant list of product requirements, a 4-step process was introduced in the Electrical and Computer Engineering (ECE) capstone design class. The 4-step approach is loosely based on The IEEE standard for requirements documents (IEEE/ANSI 830-1998 (IEEE, 1998)). This method goes beyond highlighting the importance and need for “product requirements.” It has an iterative structure (based on [2]), which uses various tasks and tools such as intra-team exercises, structured client/customer interactions and in-class workshops and various examples based on previous projects. The 4-step method is designed to be applicable to variety of projects including hardware, system design, software engineering, etc. **Figure 1** shows high-level tasks involved in each step, the tools and actions used during each step as well as input and output from each step. The 4-step approach of *Extract, Add, Quantify and Format*, thus helps the team to advance from brief project description / objective to a complete comprehensive, unambiguous, design-free list of final product requirements that is agreed upon by the team and their sponsors. The 4-step method is explained in more detail with examples in the following sections.

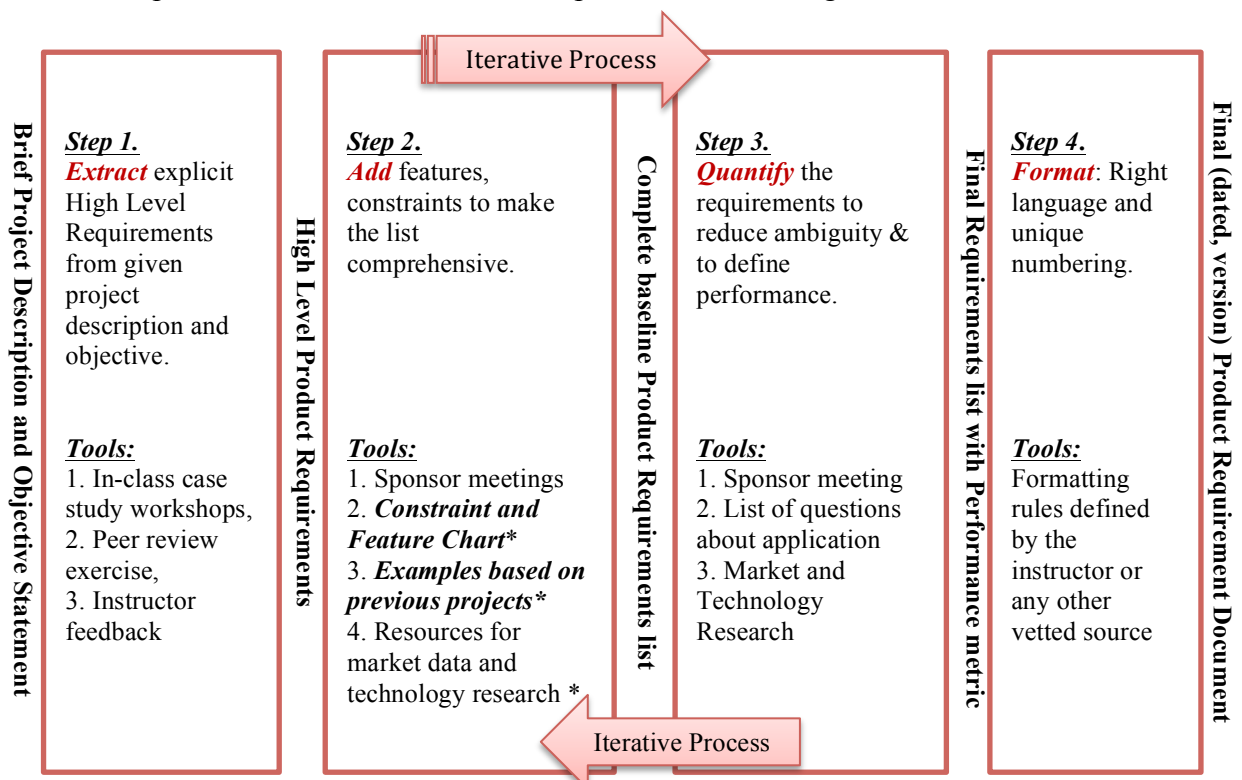


Figure 1: 4-step method describing input and output of each step and tools used in each step. Please note it may take multiple iterations of Step 2, Step 3 and sponsor meetings before Step 4. (\*Provided by the instructor)

**Step 1 (in-class, intra-team): *Extract* explicit high level requirements from the given project description and objective.**

Client companies or sponsoring entities (referred as “Sponsor”) provide brief descriptions and objectives of the projects they are offering. These project descriptions help students to choose potential projects and help the instructors (authors of this paper) to match the skills of students to the required skills of projects in order to assemble teams for each project. With this as the starting point, the team is forced to read, understand, and interpret the project’s objective/description. This step also forces the team to identify some of the obvious and explicit requirements. These in turn form the core requirements’ list for the project. The instructor typically shows a case study project example explaining Step 1 tasks, generally in an in-class workshop, so that each team can take the following actions:

1. Break down the project description, recognize and highlight key words and form a list of high-level requirements.
2. Share the list with other teams for peer review to refine the list.
3. Prepare a list of questions to obtain the instructor/s’ and sponsor’s assistance.

The high-level requirement list after this step generally captures key system behavioral features and main system functions. **Table 2** (page 8) shows the input and output of **Step 1** for a case study exercise conducted in class.

Objectives achieved during this step:

1. Team gets an easy starting point and simple tasks, which do not seem daunting.
2. Team establishes the relevance of product requirements.
3. Team gathers a list of relevant questions for sponsors to confirm their interpretation.

**Step 2 (Iterative step with sponsor interaction): *Add* features and constraints to the High Level Requirements list to make them comprehensive.**

**Step 1** develops a non-comprehensive core /basic list of requirements for the project. In **Step 2**, teams are asked to specify more functional features and product constraints to make the list more comprehensive. The goal is to describe all aspects of a product related to its operations, limitations /constraints and any underlying assumptions keeping the user / customer in mind.

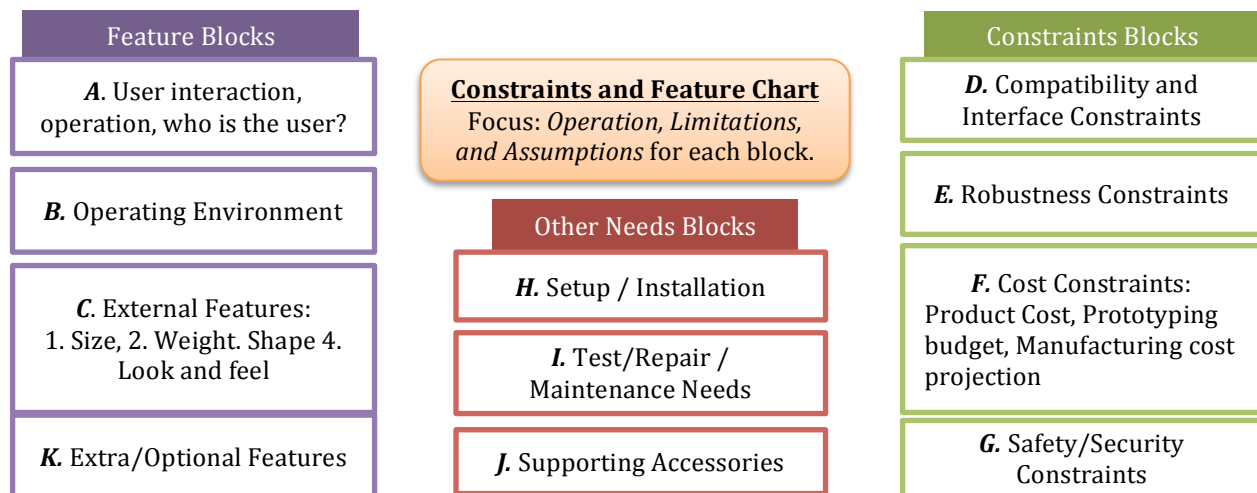


Figure 2: Above is the **Constraints and Feature Chart** explaining various aspects of the product to consider. Features are in purple. Constraints are in green and other needs are highlighted in red.

Frequently, teams, due to lack of experience, will enter a stagnant phase not knowing where to start to make the list broader and complete. The “**Constraint and Feature Chart**” (referred as **CFC**) (shown in **Figure 2**), and the relevant requirements examples from previous years are provided by the instructor to help inform and enable them to complete Step 2. For each block in the CFC, three key aspects are focused: **Operation, Limitations, and Assumptions**. Students are provided the CFC as well as the explanations and examples for each block. The example requirements come from variety of different projects. As shown below, students are provided the example projects and the respective example requirement for each block to help them think & devise questions related to their project domain.

- A. User interaction/operation: Ask who the user is and how the user will operate or interact with the system during **normal operations**. Will user have any **limitations** with the system operations? Are there any **assumptions** about user’s skills? E.g. for:
- A teaching tool: Software will be designed for easy use by middle school teachers.
  - A smart controller: User will be able to operate the system using fixed voice command.
  - A thermostat: User will be able to adjust temperature settings via a phone application.
  - An efficient power bank: User will be able to recharge the device battery using standard micro USB cable.
- B. Operating Environment: Ask what is the **normal operating** environment of the product. Product must be designed to last an expected time period in such an environment. What is that expected time period (user **assumption** about the product)? What is the shelf life (**limitation**) of the system (if applicable)? E.g. for:
- An underwater device: should operate without fail under water continuously for X hours.
  - A solar tool: Panel will operate without fail on a roof of a house for more than X years.
  - A software application: Application must run on both windows and mac OS.
  - A survey robot: Robot should traverse any kind of hard and flat surface.
- C. System size/weight/look and feel features: Ask if the product should have any size/weight/look and feel requirements based on your customer data and sponsor. E.g. for
- A location tracker: Unit should be small & portable.
  - A 3D picture booth: Booth should match the aesthetics (color and feel) of other systems on display in the library in the same space.
- D. Compatibility or interfacing constraints: Ask if the product needs to be compatible with any existing products. These features may require integration with the company’s or clientele’s other products and services. E.g. for
- A registration app: Application shall be compatible with existing open source software Y.
  - A monitoring system: Sensor platform shall not alter the existing container’s operation.
  - A data routing board: Board should interface with acquisition board v2 without external converters.
- E. Robustness Constraints: This addresses any extreme operating or shelving conditions the system will need to survive without causing permanent damage. This applies to both hardware and software projects. E.g. for:
- A multi-user app: App will allow at least 100000 users logged in simultaneously.
  - A sensor payload: System should be able to withstand (operating or not) a temperatures ranging from -40 to 85 deg Celsius.
  - A hiking location tracker: Device should operate in a forest environment (i.e., interference with tall dense trees).
  - An industrial robot: Robot shall survive a drop force of 200 lb.

- F. Cost constraints: Ask what is the projected cost of the product, if there is a limit to the prototyping budget, and/or is there a projected manufacturing volume and cost. These requirements are important because they define system design constraints. These design constraints are important to identify before the team initiates preliminary design options. Frequently, these requirements are finalized based on the sponsor's vision, product's value proposition to the customer base as well as market data. These constraints are difficult to define because they will vary based on the client's vision. Teams working on a project need to check the feasibility of a solution (e.g., is it possible to build a product) and adhere to a prototyping budget limit (e.g., product prototyping cost should not exceed \$1000).
- G. Safety / Security Constraints: Any features required to ensure safe system operation or user safety or system integrity or security, are listed here. E.g. for:
- A construction crane controller: Crane shall have a remote emergency switch to halt operation at any point.
  - An asset tracking application: Data transfer will be done over a secure channel.
- H. Setup/Installation needs: These requirements specify specific setup and installation requirements. E.g. for:
- System will not need any initial setup other than plugging in.
  - A household smart lamp: Device must be installable using common household tools.
  - A data extractor app: Software once downloaded can be installed on any Windows OS.
  - A portable tracker: System may require charging (standard 115Vac outlet) for at least x hours prior to use.
- I. Maintenance needs: Ask if the system or product is being designed to be serviceable or replaceable. If serviceable, identify what resources the users would be expected to have to maintain/service/repair the product. Will any technical support provided for the product? What is the shelf life and the operating life of the product without maintenance? E.g. for:
- In normal operating conditions, device shall not need any maintenance for x years.
  - A network tool: Device will be provided external technical support in case of failure.
  - A service robot: The following parts (TBD) shall be easily replaceable.
- J. Supporting accessories: These requirements list any extra accessories needed during system installation or operation. This may include extra documentation project requires E.g. for:
- A smart e-nose: Device shall have a printed and affixed and clearly identifiable tabulation of all chemical compounds compatible with the product.
  - A device for visually impaired: Product's buttons and knobs will have braille symbols.
  - A dishwasher: Product installation manual will be written in English / Spanish /French.
  - A smart lamp: Product will include installation hardware.
- K. Any extra features: Ask if there are additional features that the client will require or if there are any stretch (nice to have but not absolutely required) goals?

Iterative Mode: *Step 2* is performed iteratively by asking questions, negotiating solutions with sponsors, and discussing the case studies provided in class. Teams are asked to conduct a minimum of two complete cycles involving: (1) Market and technology research; (2) Intra-team discussion to draft a list of added requirements and questions for the sponsors; and (3) Sponsor discussions. Market and technology research helps identify customer needs and define the product value proposition, which can impact several of the product features.

Teams should ask questions related to each block in the CFC and add requirements accordingly. All aspects listed in CFC will not necessarily apply to all the projects. However, at this time,



teams are encouraged to address as many requirements as possible without being concerned about their importance or probability in the final product. Teams are also asked not to be concerned about repeating requirements if they fall under multiple blocks as it is preferable that no aspects of the product are overlooked. **Table 3** shows Step 2 output for the in-class case study.

**Step 3: (Iterative) *Quantify* the requirements and Define Performance**

In a product’s design process, written Product Requirements establish an agreement between various organizations (Product Management, Engineering, Marketing, Operations/supply chain, Manufacturing, Finances, etc.) for the product’s value proposition and scope. In any industry, some of the important qualities of Product Requirements are that they are *Unambiguous, Measurable, Testable, Performance Specified, Feasible, and Design-Free* [2]. Therefore, reasonable quantification of relevant Product Requirements is an essential step before finalizing the requirements. The first two steps focus on *functional* requirements of what the system does. **Step 3** focuses on *quantification* and defining *performance-based requirements* that is, how well the system performs its job. This step ensures that all the parties’ goals are aligned.

The most effective way to quantify important requirements is to focus on adjectives such as cheap, easy, small, large, portable, long-lasting, fast etc. and to ensure the requirement output is measurable and testable. Teams are encouraged to recognize and quantify these adjectives through client discussions and determining market desire. This can be achieved by asking a series of questions as illustrated in **Table 1**. If the quantification is not possible at this stage, it will be necessary to “mark” these requirements and the steps needed to address them at a later stage (See **Table 4** (page 9) for Step 3 output from case study).

Requirements needing quantification	How to quantify or explain
Cheap / Affordable	What is the customer willing to pay? What do current solutions or competitor’s products cost? How much does it cost to operate or repair?
Small / Portable	What is the application? Does this need to fit in a pocket or a backpack, a car trunk or back of pick-up? How is it powered?
Last long (Battery / power performance)	What is the normal operating environment of the device/system? What is the continuous operation? Is it in minutes or hours or days or months? Are batteries “internal” and rechargeable or are they replaceable?
Operate Fast (Speed performance)	Does user experience matter? What operational speed will solve the current problems users face with the current systems. (Speed performance)
Easy to use (light switch vs. smart phone)	What is age range/ technical expertise/ expected education level / job type/ physical skill of your average customer?
Operating environment	Think of temperature ranges / water or weather proof performance /

Table 1: Questions in relation with adjectives used in Product Requirements to further quantify as necessary.

Objectives achieved during Step 2 and Step 3:

1. Students draft questions for their sponsors to enable the creation of comprehensive list of product requirements.
2. Examine examples of case studies provide a roadmap for the process.
3. Create multiple iterations of the product requirements help reduce ambiguity between the team and the sponsor’s vision (i.e., align goals).

#### **Step 4: *Format* of the final requirements list.**

During the first 3 steps, teams will iteratively analyze the complete requirements list, discuss the list with sponsors, merge the relevant requirements, and delete redundant and repeat items / tasks before formatting the final list in *Step 4*.

Formatting rules may differ from course to course or from department to department. The main formatting requirements used in the ECE Capstone class at Author's Institute are [2][7]:

- A. Every requirement must be a complete sentence and have a unique number identifier [7].
- B. All the necessary and 'must achieve' requirements will be written as "Shall" and 'stretch goals and requirements' (i.e., nice to have) based on feasibility will be written as "May." [2]
- C. A list of keywords and acronyms used will be added to the document.
- D. Requirements shall be sectionalized under appropriate headings, E.g. "General Operating Requirements", "Added Features", "Constraints", "Optional Features" and "Other", etc.

The reason to enforce formatting requirements for the Product Requirements document is to ensure good writing practices in the beginning of a project as well as to establish an agreed formatting style for all project related documents. Formatting rule A ensures that various project entities refer to same requirement during technical discussions and testing phase. Unique requirement numbers are further used in test and verification plans to refer to certain requirements and their respective test procedures. Formatting rule B ensures clear differentiation between 'must achieve' requirements from 'stretch goals'. This will help teams focus on the appropriate preliminary design options and to plan for the stretch goals in the future.

Once the requirements are defined, quantified, and formatted, they should be reviewed and approved by the product sponsor and the team. Once the requirements are approved, the preliminary design cycle of assessing potential solutions, feasibility, performing trade-off analysis between several requirements (may vs. shall) can be initiated so that design choices can be proposed and evaluated.

To illustrate the 4-step method and its outcome, the following case study is provided.

#### **CASE STUDY:**

This is one of the case studies conducted with students in class after introducing the 4-step method by the author. The case study emphasizes the tools and tasks of the 4-step method. Generally, case studies chosen for in-class workshops fulfill at least two of the following criteria: (1) represents a successful project from industry; (2) is a project from a previous senior design project that was sponsored by a company; or (3) corresponds to a project with open-ended goals. The case study described here fulfilled criteria 1 and 3.



**Step 1: *Extract* explicit high level requirements from the given project description and objective**

<i>Input: Given objective and background description</i>	<i>Output: High level requirements after step 1</i>
<p><b>Project Background:</b> Many <b>third world countries</b> still <b>do not have electricity</b> in remote areas, which affects standard of living. The reason for this is cost as well as resource availability. Current solutions are <b>hazardous</b>.</p> <p><b>Project objective:</b> Our goal is to design and develop a <b>cheap and clean light source</b> system, which can be <b>easily used by anybody</b>. It will <b>not</b> need any <b>maintenance or running cost</b> and <b>preferably</b> work with a <b>renewable energy</b> source.</p>	<ol style="list-style-type: none"> <li>1. The system must provide light.</li> <li>2. Light source must be affordable and cheap.</li> <li>3. Light source must last long.</li> <li>4. Light source must use clean energy source and not generate hazardous outputs.</li> <li>5. Light source must not incur running cost or require maintenance.</li> <li>6. Light source may use renewable energy source.</li> <li>7. Light source must be durable and capable of operating in primitive conditions.</li> </ol>

Table 2: Example to explain Step 1 input and output

**Step 2: *Add* more features/constraints to make the list comprehensive.** Please note that this exercise was done in class. Students are asked to come up with this list for their project and iteratively discuss with sponsors before finalizing it.

A. User operation/interaction, Who is the user?	<ol style="list-style-type: none"> <li>8. Light source shall be easy to use by local villagers (possible low education level).</li> <li>9. User shall be able to turn the light using an on/off switch.</li> <li>10. User may be able to adjust the light brightness (perhaps from gentle glow to reading appropriate intensity)</li> <li>11. Light source may have a charging mode or battery mode, which can be activated by a simple button.</li> <li>12. Users may be able to put the source on any flat surface safely.</li> <li>13. Light source shall have symbols or pictures (instead of words) on operating buttons.</li> <li>14. User shall be able hang light source or mount it on a wall easily.</li> </ol>
B. Operating Environment	<ol style="list-style-type: none"> <li>15. Light sources shall operate in a dusty and hot environment</li> <li>16. Light source may operate in places with low sunlight.</li> <li>17. Light source shall be mainly used for indoor home use.</li> </ol>
C. Size/weight/look and feel features:	<ol style="list-style-type: none"> <li>18. Light source shall be portable.</li> </ol>
D. Compatibility and Interface constraints	<ol style="list-style-type: none"> <li>19. Light source may be compatible with external batteries.</li> <li>20. Light source may be able to charge other devices.</li> </ol>
E. & G. Robustness and safety constraints	<ol style="list-style-type: none"> <li>21. Light source shall survive a fall from ~2 meters on a hard ground.</li> <li>22. Source shall be enclosed properly and not have exposed wires</li> </ol>
F. Cost constraints	<ol style="list-style-type: none"> <li>23. The initial cost of the product shall be less than the current kerosene lamps but with no running cost.</li> </ol>
H. Setup and installation needs	<ol style="list-style-type: none"> <li>24. Light source shall not need any setup or configuration other than physical mounting.</li> </ol>
I. Test / Repair / Maintenance Needs	<ol style="list-style-type: none"> <li>25. Light source shall not need maintenance for an extended time.</li> <li>26. Light source shall be repairable using locally available resources such as screws,</li> </ol>

	nuts, bolts, fuses, and wires. 27. Light source shall not have any recurring cost.
J. Supporting accessories:	28. Light source shall not require any extra accessories other than mounting hardware.
K. Extra Features	29. Light Source may be aesthetically pleasing. 30. Source may be smaller than a pocket book.

Table 3: Example of Step 2 for Case Study done in-class.

**Step 3: *Quantify* some of the requirements from the list in Table 2 & Table 3**

Previous Requirement	After quantification
1. Must provide light. <i>Q: How much light is sufficient, what do current lamps provide?</i>	1. Must provide enough light for evening and night time activities such as reading, cooking, eating, etc.
2. Light source must be cheap and affordable. <i>Q. How cheap is cheap enough?</i>	2. Light source should be much cheaper than the current existing lamp systems (kerosene). Target cost \$10 - \$15
3. Light source must have a long lifetime. 26. Light source shall not need maintenance for long time. <i>Q. How long is good enough and feasible? What is the recurring cost of current kerosene lamps?</i>	Merge 3 and 26. 26. Light source shall last for > 2 years without major maintenance. 27. The actual lighting element (example LED) shall not need maintenance for at least for 2 years.
7. Light source must be durable and capable of operating in primitive conditions. 16. Light sources shall operate in a dusty and hot environment, <i>Q. What is the temperature range?</i>	Merge 7 and 16. 16. Source shall be operable in (-20C to 60C) and indoor, dusty* environment. <i>*Need to define the “dusty” based on preliminary test results to meet the prototype cost requirement.</i>
18. Light source shall be portable. <i>Q. Extent of portability?</i>	18. Light source shall be easily portable and similar to current existing light sources (to be easily carried to the needed room/place)
19. Light source may be compatible with external batteries. <i>Q. What are the standard and easily available and cheap batteries in the target area?</i>	19. Light source may be compatible with external batteries* <i>* Need to look at standard batteries easily available and most popular</i>
20. Light source may be able to charge other devices. <i>Q. What type of devices? Will it operate and charge at the same time?</i>	20. Light source may be able to charge or power other standard external devices during charging mode* (e.g. phones, flashlight, radio, etc.) <i>* The final list of possible devices will depend upon design constraints and technical feasibility.</i>

Table 4: Examples explaining Product Requirements quantification process

Step 4 (***Format***) is not always performed during an in-class case study exercise; sometimes student are encouraged to ask the questions, “what is important and what is feasible,” before making a “Shall” or a “May” decisions.

## TEST RESULTS

Authors conducted multiple surveys of senior undergraduate capstone students over the last 3 years (~200 students). These surveys were used to gauge the students' understanding of what are product requirements and how to gather them. Before introducing the 4-step method, only 10% of the students demonstrated correct knowledge about product requirements versus 80-85% after introducing the 4-step method (the same students were surveyed). The survey consisted of a series of quiz questions (See [Table 5](#) for sample questions).

One of the authors also performed an experiment to gauge the improvement in the crafting of product requirements as follows. Teams were graded on their product requirements draft after providing lecture material, other resources and examples to teach them “Preparing Product Requirements” but before teaching the 4-step method. The same teams were given an opportunity to re-do the product requirements using the 4-step method after the lecture on 4-step method and case study workshop. The same grading rubric (shown in [Table 6](#)) – provided to the teams before the first draft was due – was used for grading both the product requirements drafts. The students were not informed of the original grades or provided any specific feedback in between both the submissions. This experiment was performed with 9 project teams working on 9 different projects representing varied applications (total 36 students). Out of the 9 projects, 5 are sponsored by established companies, 2 are sponsored by start-up companies and 2 are offered by University Research Centers. Project domains include cloud software applications, wearables, IoT, robotics, circuits, and communication, and much more. [Table 7](#) (page10) shows the grading results before and after 4-step method as well as performance improvement metrics.

Sample survey and quiz question	Answer options
Have you performed exercise of eliciting “Product Requirements before? If yes when and where?	Yes, Another design class, company internship, job, other / No
How often have you interacted with project clients / project sponsor and in what capacity? If yes, please give details.	Yes / No
Product Requirements are created from whose point of view?	Customer / User / Market need / Engineer / Programmer / Marketing manager
Product Requirements focus on which project aspects? Please select or state a few.	Implementation details / technical specs / User operation / system feature /
What steps would you take to understand the product requirements of your project?	

[Table 5: Some of the questions included in the quiz/survey conducted before and after 4-step method. Some questions are multiple choice and some are input text driven.](#)

Grading aspect	Grade	Aspect Explanation
Project Background	10%	A small paragraph summarizing project background, problem statement and motivation/benefit to a customer for the product you will design.
Completeness of product requirements	50%	This includes thoroughness of the requirements, whether the requirements describe a complete product or not.
Quality of Product Requirements	30%	This includes whether the requirements are Unambiguous, Measurable, Testable, Performance Specified, Feasible, and Design-Free.
Technical Writing & Document structure	10%	This includes quality of writing, clarity of explanations & information presented and document formatting / organization.

[Table 6: The grading rubric used for Product Requirements Document](#)

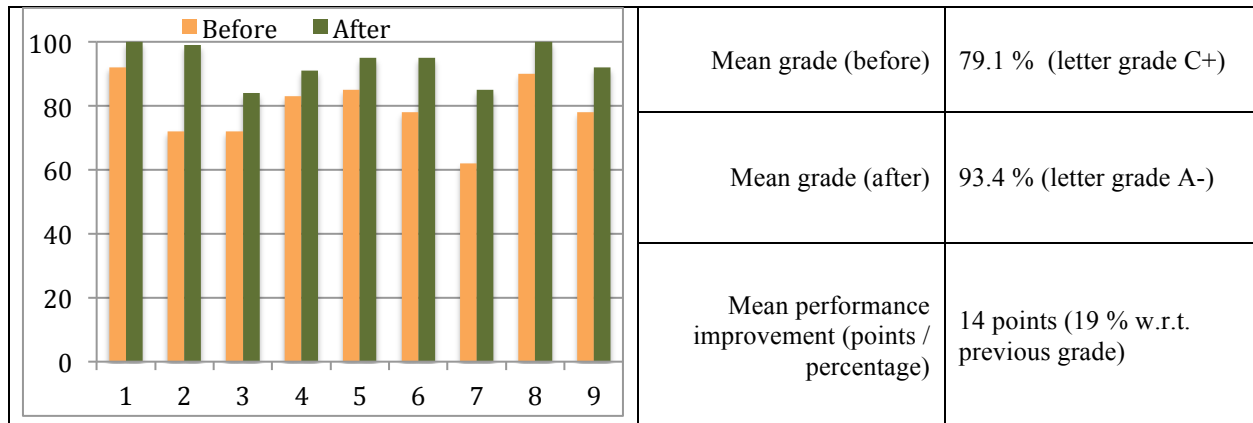


Table 7: Test Results from the grading experiment. Graph shows grades for 9 project teams before and after introducing 4-step method. 2 columns on the right hand side state the performance metrics.

## CONCLUSION

One of the authors has been teaching the ECE Capstone design class for last 6 years. The 4-step method was devised approximately 3 years ago by both the authors and has evolved to the model presented in this paper, but remains under refinement as needed. Based on the survey results, questions received from students before and after introducing the method, and sponsor feedback, it is apparent that 4-step method of **Extract, Add, Quantify and Format**, simplifies the process of gathering relevant and well-defined product requirements for senior undergraduate students irrespective of their industry exposure. The novelty of the unique 4-step method lies in the easy breakdown of the required steps into simple questions and tasks (**Extract, Add, Quantify and Format**), which can be incorporated as in-class workshops, peer review exercises, case studies of previous projects and structured sponsor meetings. Importantly, this method is applicable to multiple varieties of projects (e.g., hardware, software, feasibility study, open ended problem, etc.) and should be widely applicable outside of academia in a professional setting.

The other important observation is that the 4-step method works well when the sponsors are not sure about their “whats”. This method of realizing product requirements (because of iterative way of asking relevant questions to sponsor) not only helps the team of engineers understand the problem they are trying to solve but also helps the sponsors refine their proposition and to better understand what is realistic, feasible, or even realize the actual problem they wish to solve.

Test results presented in this paper show that all the teams benefited from the 4-step method irrespective of the project source or technology or domain.

## REFERENCES

- [1] MITRE, “System Engineering Guide: Eliciting, Collecting and Developing Requirements”, <https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/requirements-engineering/eliciting-collecting-and-developing-requirements>
- [2] MITRE, “System Engineering Guide: Analyzing and Defining Requirements”, <https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/requirements-engineering/analyzing-and-defining-requirements>
- [3] Daniels, L. “Managing the product requirements definition process.” Project Management Institute Annual Seminars & Symposium, Houston, TX, 2000.

- [4] Muci-Küchler, Weaver, "Learning How to Identify Customer Requirements: A Key Component of Product Development Courses", American Society for Engineering Education Annual Conference & Exposition, 2004.
- [5] Martin Cagan, "How To Write a Good PRD", <https://svpg.com/assets/Files/goodprd.pdf>, 2005 Silicon Valley Product Group.
- [6] Ambler, S. W., October 1, 2005, "[Requirements Wisdom](#)," *Doctor Dobb's Digest*.
- [7] Florence, A., "Reducing Risks Through Proper Specification of Software Requirements," *CrossTalk: The Journal of Defense Software Engineering*, April 2002.
- [8] Gottesdiener, E, Good Practices for Developing User Requirements©, *CrossTalk: The Journal of Defense Software Engineering.*, March 2008.
- [9] Mansour, S, [A Guide to Agile Requirements Documentation](#), Atlassian Blogs, July 10, 2013.
- [10] Bahill, T., "What Is Systems Engineering? A Consensus of Senior Systems Engineers," April 28, 2015.
- [11] MITRE, "System Engineering Guide: Requirements Engineering", <https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/requirements-engineering>