A NOVEL APPROACH TO THE EDUCATION OF POWER ENGINEERS
BUILDING THE SMART GRID

Binghui Helen Wu
FS Consulting
8209 East Brookhollow Street, Wichita, Kansas, USA
SWHELEN21@HOTMAIL.COM

ABSTRACT
The importance of the electrical energy supplies that may impact nation-wide economy has been the key to the smart grid initiative. This paper is going to pave the road leading to the urge of educating future power engineers with an introduction of the smart grid initiative, a progress report in the smart grid research, and a discussion in preparing for the future generation of smart grids. Having a unique background as a software engineer and an electrical power research engineer, the author proposes a novel approach for power engineers to be prepared for their successful career in the future of smart grid. It is a curriculum design for a power engineering program in 4-year colleges.

KEY WORDS
Education of power engineers; smart grid; power engineering curriculum; software engineering.

1. Introduction

1.1 A Review

An electrical grid is described on wikipedia.com as follows. It is an interconnected network for delivering electricity from suppliers to consumers with three main components:

1) electricity generation from combustible fuels (coal, natural gas, biomass) or non-combustible fuels (wind, solar, nuclear, hydro power),
2) electricity transmission lines, and
3) transform stations that change voltages at both ends of a transmission line so that electricity can be distributed and delivered properly.

It is essential to use computers and other technologies to modernize electrical grid in an automated fashion to improve the efficiency, reliability, security, and efficiency of the electric grid. (2) Dynamic optimization of grid operations and resources, with full cyber-security. (3) Deployment and integration of distributed resources and generation, including renewable resources. (4) Development and incorporation of demand response, demand-side resources, and energy-efficiency resources. (5) Deployment of “smart” technologies (real-time, automated, interactive technologies that optimize the physical operation of appliances and consumer devices) for metering, communications concerning grid operations and status, and distribution automation. (6) Integration of “smart” appliances and consumer devices. (7) Deployment and integration of advanced electricity storage and peak-shaving technologies, including plug-in electric and hybrid electric vehicles, and thermal-storage air conditioning. (8) Provision to consumers of timely information and control options. (9) Development of standards for communication and interoperability of appliances and equipment connected to the electric grid, including the infrastructure serving the grid. (10) Identification and lowering of unreasonable or unnecessary barriers to adoption of smart grid technologies, practices, and services.

An electrical grid is also known as a power grid. Two terms are exchangeable. In this paper, the usage of power grid is preferred.

1.2 A Proposal

The article Professional Resources to Implement the “Smart Grid” [1] by Heydt et al has brought about “the question as to where engineers needed to address the smart grid will be educated, how they should be trained, and to what levels of comprehension in integrative fields they must be educated how they should be trained, and to what levels of comprehension in integrative fields they must be educated. [1]” Because “truly implementing the smart grid initiative will take engineering professional resources of broad expertise and different profile than previously available,” Heydt’s paper has brought the needs in educating a new generation of power engineers into the spotlight.

Having been a software engineer and having designed and implemented software systems many years,
as well as having worked as a research engineer in Electrical Power Research Institute for five years, the author will propose an unconventional curriculum for educating power engineers in order to meet the need for the future of smart grids.

This paper will be organized as follows. In section two, an interpretation of smart grid will be given along with a brief survey of smart grid research, where advanced warning systems are to improve overall grid reliability [2], and the integrated requirements of the smart grid [1] are essential. Since both research directions are relying on the profound understanding and practical knowledge of information and computing technologies, a suggestion for the criteria in evaluating application software engineers will be discussed in section three. Combining both section two and three, a nonconventional curriculum for power engineering educational program emergences and it will be outlined in section four. Finally, concluding remarks will be given in section five.

2. Interpretation of Smart Grid

2.1 An Example

On the morning of July 31, 2012, North India had a massive power blackout that affected 22 of the 28 states and roughly 620 million of the total population. The worst of the blackout included the shutdown trains, stopped water delivery systems, trapped miners underground.

The Smart Grid News wrote the following report about the incident immediately: “While the specific cause has yet to be identified, the failure of three regional grids that cause the blackout has been generally blamed on state electric boards accused of over drawing power from the grid to meet heavy demand. Frankly, outages in India are so common they generally don’t attract attention. The worn out electricity infrastructure fails routinely, particularly when the country tries to meet burgeoning demand with unreliable power transmission and distribution systems.”

At the time of the report, it was too soon to learn the real reason behind the cause of that blackout. However, this blackout has provided us a valuable case study. We may ponder about the possible reasons as to the most general cause, such as the overloading in the consumer side, and about the possible preventions. This incidence is a live case study in how smart grid technologies should be brought to help.

2.2 Highly Integrated Power Systems from End to End

Heydt et al have emphasized the needs for an integrated approach to adapting advanced technologies in design and implementation of smart grids. Figure 1 portrays twelve elements involved in the engineering discipline, known as Smart Grid Engineering. Among the twelve, six engineering elements are the automatic control, information technology, power electronics, computer engineering, signal processing, transmission and distribution, which are directly influencing the quality of smart grids.

From the technological perspective, the components of these engineering elements can be categorized either as hardware or software and most hardware components have embedded software in them. We tend to call these components the smart technological components. In an old industry such as power and energy industry, making existing components smart is the prime targets of the power and energy research.

Therefore, the urge of integrations, as Heydt et al has suggested in [1], from the technological point of view, refers to applying software technologies to make systems highly coherent and effective in preparing the power grid for adapting all possible scenarios: expansions, fault prevention and detection that is interfaced with sensors, such as PMU, and actuators, such as SMT. (See definitions of SMT and PMU in 2.3.)
Madani and King [2] have studied 21 years (1984-2005) of transmission outages, attempting to come up with effective control methods such as load shedding for damping control, voltage control and transient controlling angle, where all of them can be used in a more coordinated fashion via software technologies, which are examples of smart grid research.

Adamiak, Novosel, Kasztenny, Madani, Sykes, and Phadke [3] have used Wide-Area as the term in describing the integrated power systems study, where an integrate view of systems is used to protect and control through software development. Let us look at more detailed power grid research through recent academic research publications.

2.3 Power Generation Expansion (GEP) and Grid Reliability

Meza, Yildirim, and Masud [4] have developed a long-term multi-objective model for power generation expansion planning of electric systems based on their observation of the Mexican Electric Power System and had illustrate their model solution, MGEP.

Parallel to modeling expansion, Sykes et al have studied protection systems in maintaining the stability and reliability of the electrical power grid [5]. Methods, such as Synchrophasor to monitor system conditions, have been studied in order to incorporated them into protection schemes to reduce problems stemming from a variety of hidden failure modes that has portrayed by figure 2 given in [6] and to increase the effectiveness and reliability of protection systems.

A solution proposed at the systems level in the paper Wide-area Monitoring, Protection, and Control of Future Electric Power Networks has been discussed in [7] by Terzija et al. It is to prepare for the protection of remote locations to counteract during propagation of large disturbances. Synchronized measurement technology (SMT) is an important element and the enabler of the wide-area monitoring, protection, and control (WAMPAC). It is expected that WAMPAC systems will be used to reduce the number of catastrophic blackouts and generally to improve the reliability and security of energy production, transmission, and distribution, particularly in power networks with high levels of operational uncertainties.
Figure 5 elaborates the coordination relations given in figure 4.

Clearly, the brain power of a WAMPAC system is manifested by software via advanced communication and networking technologies. Consequently, it should be implemented by software engineers, who ought to have a profound understanding of both problems and solutions of the power grids.

An effort has been taken from Software Engineering Institute of CMU, under the name of Domain Specific Software Engineering (DSSE) with the anticipation of making tools as enablers to specific domain engineering projects. The DSSE is also known as End User Programming [8].

This approach seems not as potent as one might have believed. Instead, a proposal for educating power engineers with software engineering knowledge at colleges can be an alternative.

Based on years of engineering practices, the author believes that a power engineer should have the ability to use software tools and software components that are made by themselves and for themselves.

In reality, there are inadequacy and awkwardness for software engineers to understand the problems on which domain experts have worked on for years including those challenges in the solutions they are proposing. Oftentimes, the problems are embedded in the form of software requirements specification; and they are reflected in the programs that software engineers make.

In fact, these problems go both ways. For example, while software engineers who have no clue on details of solutions to problems of a domain, domain engineers had little understandings how tools can be optimally used.

Let us look at what characters are necessary in software engineers who can work with domain engineers to develop software applications for an engineering domain.

3. Criteria of Software Engineers

Engineering as a discipline is a hands on profession applying scientific knowledge to make things working for us. Use Yoji Berra’s popularly quoted phrase as the following. In theory, theory and practice are the same. In practices, they are not.

This is true in any engineering discipline, where there are limited principles applicable to implementations of engineering solutions. In any non-software engineering curriculum, engineers are trained with scientific knowledge of their specific domains. What is the domain with which software engineers are trained? It may be Electrical, Mechanical, and Chemical, to name a few to illustrate a point. In fact, if a software engineer is to take the tasks of implementing financial software, he/she should be better off with the training in knowledge of financial dealings.

I was professionally trained as an electrical engineer; and I had worked in the power and energy industry before my graduate study in Computer Science. At the time that I chose to study in computer science, it was the sole motivation for me to learn the computing technologies to apply to the advanced controls and automations in industrial systems. That was the domain of my engineering training at my undergraduate study in a four-year engineering school.

Speaking of my own experience, during my work as a software engineer with different large scale software intensive engineering systems, the obvious benefit in getting software implemented well is my effort to understand the system for what it is and how it should be designed analytically, whether the system is in automotive, aerospace, or telecommunication. It is possible to request software engineers to be trained with a domain specific knowledge with which software is needed. But, what if we ask ourselves the following question: would it be better having a necessary training to happen the other way around? That is, we start to make a non-software engineer program to include sufficient software engineering training. Let us continue the discussion to lead to our conclusion of a non-software engineering educational program in an engineering school including an example of software applications development in the power and energy industry.

3.1 State of the Art Software Development

Software engineering as a discipline has officially been well over four and half decades [9]. Efforts in research scientific knowledge for software making have long been recorded in textbooks of software engineering: in theories, principles, and practices. [10-11] Without a doubt, useful tools for making software have been demonstrated within a very broad spectrum [12] in both technologies and methodologies, leading by the most demonstrative Agile movement: XP, Scrum, Kanban, and software craftsmanship, among many variations of Agile software development methods. In addition, the question, “Are
some programming languages better than others?" has been given in the preface of Making Software, what really works and why we believe it with an undisputed answer, "the jury is still out..." [13], even though we may find comprehensive studies of such topics to ponder [14].

The ground support of such a conclusion has been given in many software engineering literatures throughout years ever since the term “software engineering" is coined. For example, in the book Solid Software, authors have used the term “Gutless Estimation” [15] while finding reasons that software development is so hard. The term Gutless Estimation is the top reason of software development being so hard. It is, undisputedly, the main characteristic of engineering in any disciplines, including software engineering.

3.2 Importance of Domain Expertise

“A program which does not work is undoubtedly wrong; but a program which does work is not necessarily right. It may still be wrong because it is hard to understand; or because it is hard to maintain as the problem requirements change; or because its structure is different from the structure of the problem; or because we cannot be sure that it does indeed work.” These insightful commented about a program was made in 1975 [16] by M.A. Jackson. It was true then; it is true now. In fact, it could not be truer today. To elaborate Jackson’s succinct statements on failures of a program given above, the three key characteristics of programs are revealed:

1) When a program is to solve a difficult problem, the solution may be difficult to understand; therefore, it may not be obvious whether the solution is totally correct to other code readers/programmers/testers who are members of a quality assurance team.

2) A program solution is not naturally or coherently evolved with the problem, especially when problem reveals itself gradually and deeply. The solution that works initially can no longer address the problem, i.e., the program was perceived as hard to maintain or structurally different from the natural structure of the problem.

3) The complexity of the program is high and test coverage cannot be complete to ensure that the program is going to be absolutely correct.

Although modern software may be very different from those 37 years ago, when Jackson made his observations quoted above, due to the accumulative nature of our knowledge about programming and our creativities to make better programming tools for making software. The fundamental purpose of making software is still about creating solutions of problems in practical ways, whether a problem has been in existence or the problem is newly created. Therefore, understanding problems that we are applying our software engineering knowledge to the domain we worked on is the key provided that the purpose of software resides in the domain.

For example, search is a classical problem; and many different search algorithms have been well developed and implemented. The knowledge of search algorithms, along with implementations in different programming languages, has been undoubtedly the knowledge of every programmer and each software engineer. My example in [12] is an embodiment of this particular coding practice.

In that example, i.e., at every four millisecond, a large sample is received and searched. It is obviously bad programming practices if an implemented solution is not using “divide and conquer” but rather, a sequential search using a double-loop. What was wrong with people doing the sequential? Were they not thinking or not connecting between problems at hand and their knowledge learned at school? That was a typical engineering problem to recognize and to find a typical engineering solution to implement.

Software engineering as a discipline is a practical matter; we ought to recognize that among software engineers some tend to follow technologies and methodologies dogmatically, some of them compensate their drawbacks with simply do-what-the-boss-told. In fact, this is true in any engineering discipline; however, software engineering has a very low tolerance to such behaviors.

3.3 Example of WAMPAC system

As mentioned in the section two of this paper, a solution proposed at the systems level and outlined with figures 3 to 5 is used to reduce the number of catastrophic blackouts and generally improve the stability and reliability of electricity production, transmission, and distribution, particularly in power networks with high levels of operational uncertainties.

Let us inspect this system in details to discuss the needed qualification in software engineers in order to get a better design and implementation of software in WAMPAC systems.

Without the basic understanding of the PMU operational principles, it would be hard to know how the PMU data ought to be acquired anticipating relay changes to isolate local disturbances provided it is to prevent the possible disturbances in cascaded propagations within power grids.

Coordinating events requires the understanding of the cause-and-effect in these events. Consequently, it is true to come up with any acceptable, if not optimal, structural solutions in programing. An example, given in [5] shown in figure 6, is one possible coordination of PMU data with designed mechanism to achieve certain desired adjustment, i.e., an isolation effort in case of disturbances happening on power networks.

Having argued that it is necessary for software engineers to know the domain knowledge intimately in order for software applications in power grids to be successful, let us look at how curriculum design for
educating power engineering students with software engineering knowledge to achieve the goal.

4. A Curriculum for Educating Power Engineers

Heydt et al have suggested a list of 9 categories in [1] Professional Resources to Implement the "Smart Grid," as follows.

1) Direct digital control,
2) Identification of new roles of system operators,
3) Power system dynamics and stability,
4) Electric power quality and concomitant signal analysis,
5) Transmission and distribution hardware and the migration to middleware,
6) New concepts in power system protection,
7) Environmental and policy issues,
8) Reliability and Risk Assessment,
9) Economic analysis, energy markets, and planning.

Building on the top of this excellent curriculum to educate power engineers, the author of this paper proposes three more software engineering categories, in addition to above 9 categories, to complete a proposed curriculum in the training of power engineers in particular, and the training of any non-software engineering professionals in general. They are listed as follows:

1) fundamental knowledge in computer science curriculum,
2) the synchronic code studies in software, from code that enables machines to code that enables application systems,
3) the diachronic case studies in software applications crossing different industries.

In other words, it is necessary for non-software engineering students to understand software in their engineering education through the study of core courses in computer science referring 1) in the list above and the studies of code and applications, in the elements listed in 2) and 3). It may be too hard for some because engineers of modern systems need to be trained as software engineers so they can be qualified to take the tasks of working on engineering applications.

Software applications that enable engineering systems are highly in demand ubiquitously. Software engineering as a profession is in high demand. However, many studies [17-19] have indicated that it is difficult profession if not impossible to universalize it. It is true both in theorists and in practitioners among software engineering researchers.

In author's understanding, building a software application that enables engineering systems is the process of building a brain in a machine. The engineering aspect of creating the brain can be best captured with the following metaphor. That is, one should never expect Pablo Picasso (1881-1973) to request a skillful brush user to do what he is going to create in his painting.

As I have stated previously, “we tend to inference or argue via analogy. That is, we compare one particular thing that is familiar to us with another particular thing with which we want to make a point or to emphasize our point of view.” [20]

The analogy used above is by means of matching a domain expert with Picasso and a software engineer with a skillful brush user. Two abilities: knowing what to implement and knowing how to implement it, are essential in making successful software for an engineering project. It works best with one brain in one task. If it requires a team work for a large and divisible project, a relation among team members is best captured under Agile mindset.

Let me conclude the paper with a further explanation of the curriculum of the power engineering program in 4-year colleges proposed in this paper.

5. Conclusion

It is inevitable that computer (hardware) and computing technologies (software) will be heavily influencing the future of any engineering field. The software engineering part of the curriculum for the power engineering education at 4-year colleges proposed in this paper should be applicable to any other field of engineering education.

Based on the concept of Smart Grid Engineering [1], this paper gives a survey of the smart grid research. Using WAMPAC systems as the example, the paper has provided detailed discussions, an overall picture of how software engineering may be needed for the smart grid initiative to carry on. Consequently, it calls for the preparation of next generation power engineers to develop the future smart grids. Having been worked as a software engineer with hands on development of software intensive
systems, the author of this paper has provided an analysis on why it is necessary for domain experts to get involved with applying software engineering knowledge in developing software applications for the future in power supply networks, i.e., smart grids. A curriculum for power engineering education at 4-year colleges follows. The proposal is not only necessary to bring out the best of power systems modernization efforts, but also it is feasible through educating young power engineers at their college education. The three categories in software engineering education program combined with nine categories of smart grid engineering education program will be the complete and unconventional power engineer curriculum in order to prepare power engineers to work on the future smart grids.

References

[14] L. Prechelt, Two comparisons of programming languages, Making software, what really works and why we believe it (O'Reilly, 2011,) Chapter 14, editors, Oram and Wilson, p239-258.
[17] J.E. Hannay, Personality, intelligence, and expertise: impacts on software development, Making software, what really works and why we believe it (O'Reilly, 2011,) Chapter 6, editors, Oram and Wilson, p79-110.
[18] A. Begel and B. Simon, Novice professionals: recent graduates in a first software engineering job, Making software, what really works and why we believe it (O'Reilly, 2011,) Chapter 26, editors, Oram and Wilson, p495-516.