Holistic Development of Computer Engineering Curricula Using Y-Chart Methodology

Muhammad Rashid and Imran A. Tasadduq

***Abstract*—The exponential growth of advancing technologies is pushing curriculum designers in computer engineering (CpE) education to compress more and more content into the typical 4-year program, without necessarily paying much attention to the cohesiveness of those contents. The result has been highly fragmented curricula consisting of various disconnected theory courses and laboratory practicals. A holistic approach to curricula development that focuses on integrated fundamental knowledge is required to remedy this. This paper proposes a holistic approach to developing a CpE curriculum and to maintaining continuous, coherent development of that curriculum. The elements of the proposed approach are the following: formulation of objectives, identification of major curriculum areas, construction of a system model for CpE core courses by using the Y-chart methodology, description of electives, integration of laboratory practices, and assessment of the curriculum. An assessment of the curriculum is presented based on survey data and accreditation outcomes. Assessment results show that holistic understanding and integrated design experience assist students in achieving the program objectives.**

***Index Terms*—ABET self-study, forward and backward references, integrated design experience, lifelong learning, program objectives, student outcomes, system model.**

I. Introduction

CONTRIBUTIONS in information technology and communications have resulted in revolutionary technological advances [1], [2]. Computer engineering (CpE), as with other disciplines, is being driven by this exponential progress in applications and technology [3]. To cope with this technological explosion, the 2004 IEEE/ACM curriculum provided fundamental guidelines for undergraduate CpE degree programs [4]. Several other CpE curricula have also been rationalized [5]–[16]. Although these efforts represent a significant contribution, there remains a fundamental problem of meeting educational needs within the constraint of a 4-year program.

In conventional CpE curricula [5]–[16], attention is mainly given to individual courses; the integrated delivery of knowledge and design experience is often overlooked. The design of these individual courses often leaves unclear their contribution to the curriculum as a whole. Laboratory practicals are designed in a discrete relationship to the course in which they are offered, rather than coherently across the curriculum. Consequently, these collections of courses and laboratory practicals cannot provide an overall vision of CpE.

This paper proposes a holistic approach to developing a CpE curriculum and to maintaining continuous, coherent development of that curriculum. The proposed methodology is illustrated by the computer engineering courses at Umm Al-Qura University (UQU), Mecca, Saudi Arabia. This approach can easily be applied to other disciplines, such as electrical engineering, computer science, and so on, with only minor changes. In the holistic approach, the curriculum is viewed as follows.

Definition 1.1: Holistic Vision of a Curriculum: A curriculum serves as a system to merge all the technical evolutions in a particular field into a unified course of study, forming an abstract but structured development cycle that proceeds from the concept to the final design experience.

The process of developing a curriculum starts by formulating a set of objectives, which must remain viable for some time to come. Based on these objectives, major curriculum areas are identified. In the CpE domain, the three major areas are general education, basic sciences and mathematics, and specialty CpE courses.

This paper focuses only on the CpE courses, which are further classified into computer engineering core (CEC) courses and computer engineering elective (CEE) courses. A holistic understanding of the CEC courses is essential to focus on basic knowledge, training, and tools. The CEC courses are holistically related in a system model constructed using the Y-chart methodology [17], [18]. The curriculum starts by presenting component-level courses and then merges into system-level courses. Component-level courses focus on two main components of a computer system: hardware and software. System-level courses then go on to provide an integration of hardware and software. The system model of CEC courses holistically binds the component-level and system-level courses by providing forward and backward references.

A forward reference highlights the significance of a current topic by indicating how it will be used in future courses. A backward reference takes already-acquired knowledge (such as a piece of code, a circuit model, or a formula) from a previous course and uses this material as a foundation on which to build more knowledge. A holistic understanding in the CEC courses provides the foundation and pathway to learning in the CEE courses when necessary or desirable. CEE courses provide advanced knowledge in one or more areas.

Student understanding in CpE courses can be strengthened by providing integrated laboratory practicals. For this to be effective, horizontal and vertical integration of these practicals is essential. Horizontal integration describes the relationship

Manuscript received November 04, 2012; revised February 19, 2013; July 19, 2013; and November 19, 2013; accepted January 25, 2014. Date of publication February 24, 2014; date of current version July 31, 2014.

The authors are with the Department of Computer Engineering, Umm Al-Qura University, Mecca 21955, Saudi Arabia (e-mail: mfelahi@uqu.edu.sa; iatasadduq@uqu.edu.sa).

Digital Object Identifier 10.1109/TE.2014.2304930

of individual laboratory practicals with the corresponding theory course. Vertical integration describes the integrated development of design experiences and their culmination in the final capstone design experience. The last stage of the proposed methodology is assessment, based on student surveys and accreditation outcomes.

The rest of this paper is organized as follows. Sections II and III focus on the pressing concerns in curricula development in the CpE domain and related work. The proposed holistic approach for CpE curricula is presented in Section IV. To illustrate the viability of the proposed approach, Section V summarizes the assessment results. Finally, Section VI concludes the paper.

II. Challenges of Curriculum Design

This section describes the challenges associated with curriculum development in the CpE discipline.

A. Rapid Evolution of Technology

The scope of the CpE domain has changed dramatically over the past several decades. This rapid evolution of computer engineering requires continual review of the CpE curriculum. The professional associations in the CpE discipline (IEEE and ACM) provided guidelines in 2004 [4], but given the pace of change in the discipline, updating the curriculum once a decade has become an unworkable frequency.

B. Multidisciplinary Advances

In addition to the rapid evolution of technology, CpE domain draws its foundations from a wide variety of disciplines [16]. Due to these multidisciplinary advances, a typical CpE graduate may switch areas a few times during his or her career. As advances become increasingly multidisciplinary, it is becoming increasingly difficult within a 4-year curriculum to cover the material that industry demands from its design engineers.

C. From Programs and Circuits to Systems

Rapid multidisciplinary technological advances have driven industry to embrace system-level design, in which the physical is married to the abstract [2]. Academia is developing new curricula that support a broader approach to the CpE domain [12]. Consequently, CpE students should enhance their understanding of the relationships between parts of a heterogeneous design by focusing on systems.

D. Courses as Discrete Building Blocks

In traditional curricula, courses are offered as discrete building blocks, with many instructors not even knowing the entire curricula within which their course sits and what is being taught in the other courses. Designing a curriculum is like building a wall: Rough stones cannot simply be stacked one upon another. Stones must be shaped to fit each other, and other materials must be used to bind them together. Similarly, courses should be designed to fit together and to integrate into a cohesive program.

E. Scattered Design Experiences

As with theory courses, laboratory practicals are designed in isolation rather than being related to ensure integrated development of design skills [19], [20]. The laboratories should be integrated horizontally (relationship of a laboratory practical with the corresponding theory course) as well as vertically (development of the design experience from the basic concept to the final capstone design experience). This implies that laboratory courses should be carefully distributed to deliver an integrated design experience [14].

Section III describes the state of the art in addressing these challenges and details the novelty of the work presented here.

III. Related Work

A history of the evolution of CpE curricula from the first effort in 1968 [5] to the IEEE/ACM guidelines in 2004 was discussed in [4]. Since then, several efforts have been made to revise the CpE curricula. Section III-A discusses only those efforts that call for a holistic approach. The innovative points of the proposed work in this paper are described in Section III-B.

A. Literature Review

A generic curriculum model was presented in [10] to highlight the conceptual foundations, but its generic guidelines were too abstract to be realized into a coherent curriculum without significant effort. Another similar set of guidelines, called integrated learning, was described in [7], but again a holistic curriculum based on these guidelines was never proposed.

A curriculum for an Electrical and Computer Engineering (ECE) program was proposed in [11]. Its emphasis was on a well-designed liberal education, a solid grounding in fundamentals, a wide-ranging problem-based education, and a hands-on learning experience. It remains, however, an assemblage of discrete courses without any cohesiveness to form an integrated program. Furthermore, it offers no integrated approach to imparting design skills.

An integrated approach to delivering design skills through a 3-year-long design project is presented in [6]. The students are exposed to industrial tools in the course of their studies. The limitations of this approach are that a lot of student time is spent mastering the computer automated design (CAD) tools and design methods, and a large infrastructure may be required before implementing the curriculum.

The work in [13] embeds the ideas of integrated learning from [7] with the IEEE/ACM guidelines [4]. Explicit forward and backward references between courses are used to establish relationships between courses. However, this work ignores design skills obtained in the laboratory. Another approach for the development of engineering curricula that has a specific focus on student design skills is presented in [14]. The curriculum is divided into three stages: structured design experience, guided design experience, and open-ended design experience. The objectives, implementation mechanism, and expected outcomes of each stage are outlined.

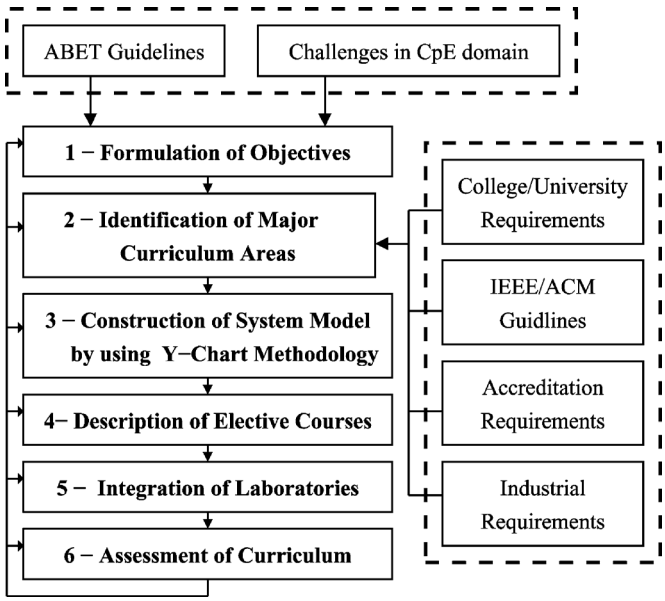

Fig. 1. Proposed curriculum methodology in the CpE domain.

B. Novelty of the Proposed Holistic Approach

The novelty of the approach can be summarized as follows.

1) The existing literature on engineering curricula acknowledges the importance of holistic understanding, but does not provide a holistic curriculum for any engineering discipline. The approach proposed here is the first effort to do this.
2) The work in [13] focuses on the coherency of theory courses, while [14] describes the integration of laboratory practicals. The approach proposed here is holistic in that it integrates theory courses and laboratory practicals.
3) IEEE and ACM guidelines [4] and ABET accreditation requirements [21] do not ensure the integrated delivery of contents. The proposed approach describes the integrated delivery of engineering education and design experiences by using the Y-chart methodology [17], [18].
4) While the Y-chart methodology is becoming standard in electronic design automation and the design of computing systems, this paper incorporated it into curriculum design.

Section IV highlights the salient features of the proposed holistic approach for the curricula development.

IV. Proposed Curriculum Methodology for Computer Engineering

The proposed holistic approach to curriculum development for the CpE domain, shown in Fig. 1, consists of six steps: formulation of objectives, identification of major curriculum areas, construction of a system model for the CEC courses, description of the CEE courses, integration of laboratory practicals, and finally an assessment of the curriculum. Sections IV-A–IV-D describe each step in detail.

A. Formulation of Program Educational Objectives (PEOs)

A curriculum must be designed to achieve a set of objectives [11]. Remembering, first, the ABET guidelines for the formulation of objectives and, second, the challenges associated with the curriculum development described in Section II, the following four objectives are expected to remain valid for some considerable time to come in the CpE domain:

1) to provide students with an education that allows them to develop into responsible citizens;
2) to provide students with a set of fundamental knowledge and skills useful for self-motivated learning;
3) to provide students with a holistic foundation and background in the CpE to make them productive engineers;
4) to provide students with an integrated design experience such that they can think in terms of systems.

Furthermore, ABET has defined certain student outcomes (SOs), identified as criteria a–k as shown in Table I. A mapping of PEOs to SOs is shown in Table II.

TABLE I
ABET a–k Criteria on Student Outcomes

Criteria	Description
a	An ability to apply knowledge of mathematics, science and engineering
b	An ability to design and conduct experiments, as well as to analyze and interpret data
c	An ability to design a system, component, or process to meet desired needs
d	An ability to function on multidisciplinary teams
e	An ability to identify, formulate, and solve engineering problems
f	An understanding of professional and ethical responsibility
g	An ability to communicate effectively
h	A broad education necessary to understand the impact of engineering solutions in a global and societal context
i	A recognition of the need for, and an ability to, engage in life learning
j	A knowledge of contemporary issues
k	An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

TABLE II
Mapping Program Educational Objectives on Student Outcomes

Program Educational Objectives (PEOs)	Measured by Student Outcomes (SOs)
1	f, g, h, i, j
2	a, b, g, h, i, j
3	a, b, c, e, f, h, i, k
4	b, c, d, e, k,

B. Identification of Major Curriculum Areas

Once the PEOs have been defined and mapped to SOs as shown in Table II, the next step is to identify the major curriculum areas. Fig. 1 shows that major curriculum areas can be identified with the help of IEEE/ACM guidelines, industry requirements, accreditation requirements, and college/university requirements. Considering the guidelines from all the sources, three major areas (A, B, C) can be identified.

1) Curriculum Area A: General Education: This area covers the basic skills (such as communication skills, humanities, or engineering economics) required by the college or university and by industry for a CpE program.

2) Curriculum Area B: Math and Science: ABET specifically states that the curriculum should teach at least 32 credit hours of mathematics and basic sciences.

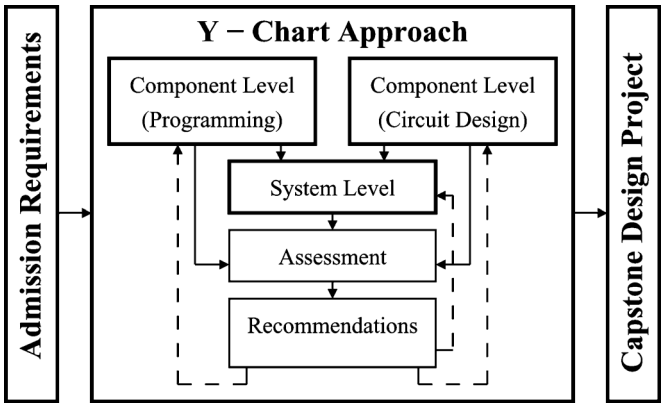

Fig. 2. System model of computer engineering core courses: abstract view.

TABLE III
PROGRAM EDUCATIONAL OBJECTIVES, CORRESPONDING STUDENT OUTCOMES, AND MAJOR CURRICULUM AREAS

PEOs	Measured by SOs	Covered in Curriculum Area
1	f, g, h, i, j	A,B
2	a, b, g, h, i, j	A,B,C
3	a, b, c, e, f, h, i, k	C
4	b, c, d, e, k,	C

3) Curriculum Area C: Computer Engineering Courses: There are two types of engineering courses: 1) CEC, and 2) CEE. CEC courses focus on the basic knowledge, training, and tools needed to learn more advanced or related topics. The second types of engineering courses are the CEE courses. Although the IEEE/ACM guidelines provide some general comments, there is no hard boundary between CEC and CEE courses.

Consequently, Table III indicates the relationship between program objectives (1–4), outcomes (a–k), and major curriculum areas (A–C).

C. Construction of System Model for CEC Courses

According to Definition 1.1, in the holistic approach, a curriculum is viewed as a system to mold all the technical evolutions in a field into a unified course of study that proceeds from fundamental concepts to a final design experience. Section IV-B identified the probable areas of curriculum in the CpE domain. This paper only considers the CEC courses, focusing on the engineering of computing systems. An abstract view of the system model for the CEC courses with the help of a Y-chart [17] is shown in Fig. 2.

The Y-chart methodology is used in the design of electronic systems and is well documented in [17], [18]. The basic concept embodied in the Y-chart methodology is the concurrent development of hardware and software. Application and architecture are separately specified at different levels of abstraction; they are then mapped to bridge the gap between the application and the target architecture for that application in the presence of some input parameters. In this paper, the Y-chart methodology is applied to curriculum design. The input to the curriculum design process is the minimum admission requirements for the CpE program, and the output is the final design experience in the form of capstone design project.

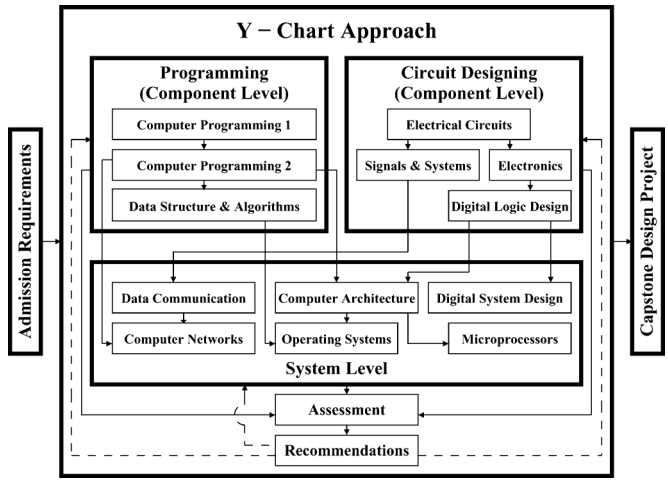

Fig. 3. System model of computer engineering core courses: detailed view.

According to the IEEE/ACM curriculum guidelines, the roots of CpE lie in circuit design and programming. Therefore, beginning with fundamental courses in circuit design and programming, and progressing through a sequence of system-level courses, students are exposed to CpE concepts that build upon one another to culminate in a senior-level capstone design experience.

Courses related to circuit design and programming are termed component-level courses because there is no integration of hardware and software in these courses. On the other hand, system-level courses provide a meaningful integration of hardware and software. Assessments are performed to generate recommendations for component-level as well as system-level courses (shown by dotted lines in Fig. 2). A detailed view of the system model for the CEC courses is shown in Fig. 3.

1) Component-Level Courses (Circuit Design): The CpE domain is mainly concerned with computing systems, whose basic building blocks are logic gates. A logic gate itself is made up of electrical and electronic components. Therefore, an understanding of electrical and electronic circuits is essential. Circuit analysis techniques for passive components (resistors, capacitors, and inductors) are studied in Electrical Circuits and are used to understand active components (diodes and transistors) in Electronics. In other words, Electronics provides backward references to Electrical Circuits. By using diodes and transistors, logic gates are implemented, thereby making forward references to the Digital Logic Design course. Finally, the Signals & Systems course describes the fundamentals of manipulating signals and thus provides forward references to the Data Communication course.

2) Component-Level Courses (Programming): The programming courses Computer Programming-1, Computer Programming-2, and Data Structure & Algorithms present data structures, algorithms, and problem solving using high-level languages (HLLs) and teach students the significance of programming within computer engineering. These courses provide forward references to many system-level courses such as Computer Architecture and Microprocessors. They help students to appreciate the need for good programming skills and to differentiate between hardware and software engineers.

3) System-Level Courses: The study of system-level courses starts with the Digital System Design and Computer Architecture courses. This is the point at which the two elements, hardware and software, combine. They use forward and backward references to other courses in the curriculum to help students tie concepts together. For example, backward references include various digital components such as the clock, logic gates, or combinational and sequential circuits, previously discussed in the Digital Logic Design course. Forward references can be made to capstone design projects to help students understand the significance of the material under study. While Computer Architecture and Digital System Design are the initial courses that combine hardware and software, it is in the Operating Systems course that students come to understand the interdependence of computer hardware and software. Similarly, the Microprocessor course is concerned with the interface between the processor and the peripherals.

A computing system consists of computations and communications. The system-level courses discussed so far are concerned with the computations part, but the communications part is equally important. The Data Communication course is usually the first course in the area of communications and networks and mainly covers the physical-layer aspects of a typical communication system or a network. Fundamentals of representing and manipulating signals are drawn from the Signals and Systems course, while the fundamentals of digital signals are drawn from the Digital Logic Design course. Forward references are made to the Computer Networks course in which students learn about the higher layers of a typical network. The Computer Networks course may also include programming projects to let the students know how two or more computers can be made to communicate through software.

This section has described how forward and backward references bind the component-level and system-level CEC courses, providing students with a solid grounding in topics across the the CpE domain. Advanced knowledge in one or more areas of the CpE domain is provided through the CEE courses, described below.

D. Elective Courses

There are several possibilities for elective courses within a CpE curriculum. For example, a CpE program could have two main tracks: Computer Systems Engineering and Computer Networks. In addition to elective courses within these tracks, there may also be free elective courses not linked to any track. In the Computer Systems Engineering Track, typical courses include Digital Electronics, Principles of VLSI Design, Advanced Computer Architecture, Embedded Systems, and Parallel Computing. In the Computer Networks Track, typical courses include Network Design & Management, Mobile Computing, Computer & Network Security, Wireless Networks, and Wireless Communications. The electives, whether in a track or free, may or may not include laboratory practicals, depending upon the program requirements and infrastructure.

E. Integration of Laboratory Practicals

Section IV-C described the system model of CEC courses holistically by providing forward and backward references. This section focuses on the integrated delivery of CEC laboratory practicals. Two types of laboratory integration have been identified: vertical and horizontal.

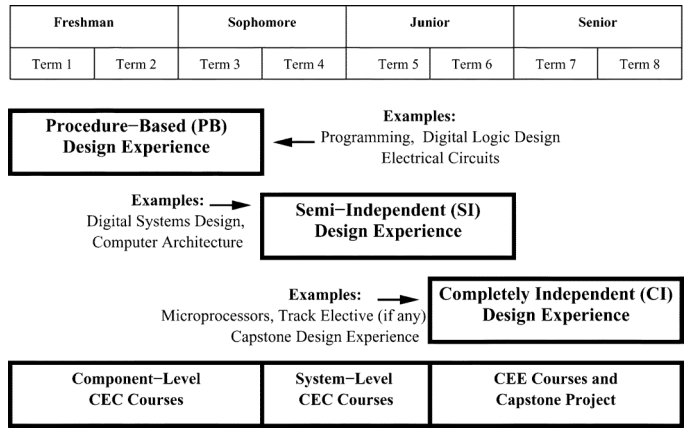

Fig. 4. Three stages of vertical integration in terms of academic years.

1) Vertical Integration: This is defined as the continuous and coherent development of engineering students' design skills throughout the entire undergraduate curriculum. In order to ensure vertical integration, three stages have been identified and distributed over the four academic years as shown in Fig. 4. These three stages are the following: procedure-based (PB) design experience, semi-independent (SI) design experience, and completely independent (CI) design experience.

In terms of study years, the freshman year focuses on procedure-based design experience, whereas the sophomore year provides either a procedure-based (Term 3) or semi-independent (Term 4) design experience. Similarly, the first half of the junior-year (Term 5) delivers semi-independent design experience, while the second half (Term 6) provides semi-independent as well as completely independent design experience. Finally, senior-year courses only provide completely independent design experience.

a) Procedure-Based Design Experience: This involves comprehensive and well-defined step-by-step procedures and instructions. The main outcome of this stage is an understanding of engineering design practice through hands-on experiments and simulations. Generally, the component-level courses (circuit design and programming) provide the procedure-based design experience. The circuit design related laboratories are in the Electrical Circuits and Electronics courses, so their objectives range from illustration of basic electrical and electronic circuit concepts to implementation of logic gates by using active and passive components. Similarly, adequate programming skills are very important for computer engineers if they are to experience system design.

b) Semi-Independent Design Experience: The objective of this phase is to start exposing students to hands-on design experience, in contrast to procedure-based design in which they are guided by well-defined procedures. These semi-independent laboratory practicals are found in courses such as Digital System Design and Computer Architecture. In the former, a computer system is designed at the Register Transfer Level (RTL) by using an HDL language (VHDL or Verilog). In the latter, the

same computer system is designed using a System Level Design (SLD) tool such as the Processor Designer from Synopsys.

c) Completely Independent Design Experience: This stage is the highest level of design experience in a students' engineering education and is delivered through the CEE courses and capstone design project. At this stage, students should take all the design process decisions completely independently, with virtually no guidance. A number of the capstone projects may come from industry, providing students with the opportunity to work on real engineering problems, using industry-standard engineering tools.

2) Horizontal Integration: This describes the relationship between a laboratory practical and the lecture (providing the theory behind that practical). There are three different forms: complete independence, dependence across terms, and close coupling.

a) Complete Independence: The laboratory practical may be completely independent from the course in which the corresponding theory is delivered. Examples might include experiments related to laboratory equipment (trainers, oscilloscopes, multimeters, and function generators) or the working environment of any software tool used to conduct experiments.

b) Dependence Across Terms: In this format, the theory is covered during one term, and the laboratory course is taken independently in another. For example, the Microprocessors course teaches the programming knowledge for microprocessor-based design in one term, but the projects based on this programming knowledge may be carried out in the following term.

c) Close Coupling: In this format, the theory lecture provides theoretical foundations for subsequent laboratory practical. For example, the concepts of half- and full-wave rectification are explained in a theory lecture, and a subsequent session in the Electronics laboratory contains experiments based on the half- and full-wave rectification concepts.

V. Curriculum Assessment

The final step in the proposed curriculum methodology is assessment, as indicated in Fig. 1. This section will show how various data were used to assess the curriculum. Student survey data were used to evaluate student satisfaction and self-confidence, and the ABET self-study was used to develop and measure the student outcomes.

Because forward and backward references holistically bind every course in the methodology, every single course, whether component-level or system-level, is being taught as a *part* of the *whole*. Consequently, the students whose assessment data are reported here had met the concepts embodied holistically in the curriculum many times in various courses. The result was a significant improvement in their results, as can be observed from the student survey data (Section V-A) and the ABET self-study (Section V-B).

A. Student Surveys

For the last 2 years (2011 and 2012), a survey has been administered to final-year students in the CpE Department at Umm Al-Qura University to assess their satisfaction with the program as shown in Table IV. They were asked to rate the extent to which they felt they had met each SO on a 5-point scale: $1 = \text{very poor}, 2 = \text{poor}, 3 = \text{good}, 4 = \text{very good}$, and $5 = \text{excellent}$. For each SO, the weighted average was calculated as a percentage. For example, in 2011, 13 out of the 23 student respondents rated outcome "a" as "good," with the other 10 students rating it "very good." None of the students replied "very poor," "poor," or "excellent." The percentage weighted average for outcome "a" is calculated as $[(1 \times 0 + 2 \times 0 + 3 \times 13 + 4 \times 10 + 5 \times 0)/23] \times 100/5 = 68.7\%$. The percentage weighted averages of student responses to the SOs for academic years 2011 and 2012 are shown in Table IV.

TABLE IV
Percentage Weighted Average of Students' Responses to the SOs, Measured at the End of Each Academic Year

Student Outcomes	2011 (N = 23)	2012 (N = 23)
a	68.7%	78.3%
b	61.7%	79.1%
c	65%	73%
d	61.8%	75.7%
e	62.6%	79.1%
f	63.6%	75.7%
g	65.2%	75.7%
h	68.2%	76.5%
i	67.9%	74.8%
j	67%	73.9%
k	60%	78.3%

The holistic approach was implemented in 2011, so students who graduated in 2011 (first column, Table IV) had experienced the proposed methodology in their courses for only 1 year. The students who graduated in 2012 (second column, Table IV) had followed methodology for 2 years. The significant improvement from 2011 to 2012 in student responses for the SOs (from "a" to "k") indicated the students' increased level of satisfaction.

The outcomes "b" (an ability to design and conduct experiments, as well as to analyze and interpret data), "d" (an ability to function on multidisciplinary teams), and "k" (an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice) were mainly covered during laboratory practicals. Consequently, improvements in these outcomes are due to the concepts of horizontal and vertical integration of laboratories presented in Section IV. Considerable improvements can be seen in the outcomes "c" and "j." However, better results are expected in future years as the concepts of system design and contemporary issues will be further strengthened through the use of forward and backward references in the methodology.

B. ABET Self-Study

Any program applying for the ABET accreditation prepares an ABET self-study report to demonstrate its compliance with the relevant ABET criteria and policies. The self-study forms the initial basis upon which the review team judges whether the program meets the ABET requirements. A self-study report is expected to be a quantitative and qualitative assessment of the strengths and limitations of the program being reviewed [21], [28].

TABLE V
ASSESSMENT RESULTS FOR DIGITAL LOGIC DESIGN COURSE

Term	a		b		c		e	
	M(a) %	P(a) %	M(b) %	P(b) %	M(c) %	P(c) %	M(e) %	P(e) %
Spring 2012 N = 18	29.5	37	29.5	37	16	0	15	56
Fall 2012 N = 18	38.5	51	38.5	51	8	67	10	67
Spring 2013 N = 15	54.2	47	25	99	10.8	55	NA	NA

TABLE VI
ASSESSMENT RESULTS FOR DIGITAL SYSTEM DESIGN COURSE

Term	a		b		c		d	
	M(a) %	P(a) %	M(b) %	P(b) %	M(c) %	P(c) %	M(d) %	P(d) %
Spring 2012 N = 15	29.5	37	29.5	27	16	55	15	26
Fall 2012 N = 38	33.3	60	8.3	29	45	73	8.3	29
Spring 2013 N = 30	28.3	60	8.3	50	50	63	8.3	50

Demonstrating the impact of the proposed methodology does not require several promotions of students to first complete their studies under the methodology. The methodology's continuous improvement procedure is mainly based on whether ABET SOs are being met in several courses. The metric for a given SO is the percentage of students obtaining a prescribed level of success in direct assessments, such as quizzes, homework, and examinations. SO satisfaction criteria are usually stated by the programs as "X% students obtaining Y% or more marks in an SO." Students encounter the holistic view of the curriculum in many courses. To show the methodology's impact two of these will be taken as a case study; these are the important courses Digital Logic Design and Digital System Design.

Digital Logic Design merges programming and circuit design courses. It provides several backward references to other component-level courses, for example how logic gates are implemented by using diodes and transistors. Similarly, it provides several forward references to other system-level courses, for example various digital components such as clock, logic gates, and combinational and sequential circuits are discussed in Digital Logic Design and then reused in other system-level courses. Similarly, Digital System Design provides several forward and backward references to other courses in the curriculum to help students tie concepts together.

The assessment results for Digital Logic Design are given in Table V, in which each row reports assessment data for one term. The first column gives the term under assessment, and the number of students (N) enrolled in the course. The remaining columns display assessment results for four SOs ("a," "b," "c," and "e") assessed in this course. Each SO is further divided into two columns. For example, the SO "a" has two columns, M(a) and P(a). M(a) is the marks that were allocated to questions used in the assessment of SO "a." P(a) is the percentage of students who achieved 70% or higher marks in SO "a." Marks allocated to questions addressing a particular SO vary semester to semester depending upon the instructor teaching the course and his/her priorities. As a result, values under the M(x) column are not identical for the three semesters. In Spring 2012, 29.5% marks were allocated to the questions that were used to assess SO "a," and 37% students earned 70% or higher marks in these questions.

To assess the methodology's impact on SO "a," see column P(a). It can be seen that SO "a" shows improvement over the three terms with 37% of students fulfilling the satisfaction criterion in Spring 2012, 51% in Fall 2012, and 47% in Spring 2013. Similarly, to assess the impact on SO "b," see the column P(b). The percentage of students who fulfilled the satisfaction criterion in Spring 2012 was 37%, in Fall 2012 it was 51%, and in Spring 2013 99%. The impact on SO "c" was that in Spring 2012, no student was able to achieve 70% or higher. In Fall 2012, 67% of students, and in Spring 2013, 55% of students achieved the satisfaction criterion. SO "e" was assessed in two terms only: Spring 2012 and Fall 2012; P(e) was 56% in Spring 2012, and 67% in Fall 2012. In Spring 2013, the instructor who taught this course did not assess SO "e."

There is thus a marked improvement in SO "b" over the three terms, and some improvement in SO "e" over the two terms it was assessed. There is a substantial improvement in SO "c" from 0% in Spring 2012 to 67% in Fall 2012. SO "a" first shows improvement over two terms, then marginally declines in the last term.

Table VI mirrors Table V, taking the case of the Digital System Design course, for which the SOs assessed were "a," "b," "c," and "d." Digital System Design is an advanced-level course that requires prior knowledge from Digital Logic Design. Columns P(a), P(b), P(c), and P(d) of Table VI demonstrate even more strongly the positive impact of the methodology. Over the three terms assessed, SOs "a," "b," and "d" show consistent improvement in the percentage of students who achieved the satisfaction criterion of earning 70% or more marks. SO "c" shows improvement from Spring 2012 to Fall 2012, and then marginally declines in Spring 2013. However, P(c) is still better in Spring 2013 than in Spring 2012.

VI. CONCLUSION

This paper has presented a holistic approach for the development of CpE curricula. Students encounter individual courses and design practices in the context of the entire curriculum. The process starts by formulating a set of objectives, based on which major curriculum areas are identified. CEC courses were identified as such an area, and a system model for these courses was constructed using the Y-chart methodology. The system model divided the CEC courses between the component- and system-level courses, and forward and backward references were provided to bind them holistically. The CEC concepts were further strengthened by providing horizontally and vertically integrated design experiences. Assessment was made via student surveys and the ABET self-study. Employer feedback in years to come will further show the impact of the methodology and

will be used to measure student abilities such as communication skills, problem solving, ability to work in teams, structured thinking, creative thinking, lifelong learning for continuous improvement, and professional responsibility. In the meantime, the work reported in this paper is intended to help curriculum designers cope with the emerging challenges.

REFERENCES

[1] J. Sifakis, "A vision for computer science—The system perspective," *Central Eur. J. Comput. Sci.*, vol. 1, no. 1, pp. 108–116, 2011.

[2] J. Teich, "Hardware/software codesign: The past, the present, and predicting the future," *Proc. IEEE*, vol. 100, no. 13, pp. 1411–1430, May 2012.

[3] A. McGettrick, M. D. Theys, D. L. Soldan, and P. K. Srimani, "Computer engineering curriculum in the new millennium," *IEEE Trans. Educ.*, vol. 46, no. 4, pp. 456–562, Nov. 2003.

[4] The Joint Task Force on Computing Curricula: IEEE Computer Society and Association for Computing Machinery, "Curriculum guidelines for undergraduate degree programs in computer engineering," 2004.

[5] W. F. Atchison, S. D. Conte, J. W. Hamblen, T. E. Hull, T. A. Keenan, W. B. Kehl, E. J. McCluskey, S. O. Navarro, W. C. Rheinboldt, E. J. Schweppe, W. Viavant, and D. M. Young, Jr., "Curriculum 68: Recommendations for academic programs in computer science: A report of the ACM curriculum committee on computer science," *Commun. ACM*, vol. 11, no. 3, pp. 151–197, 1968.

[6] A. K. Uht, "The integrated computer engineering design (ICED) curriculum," in *Proc. Workshop Comput. Archit. Educ.*, 1998, Art. no. 4.

[7] J. D. Mccowan and C. K. Knapper, "An integrated and comprehensive approach to engineering curricula, part one: Objectives and general approach," *Int. J. Eng. Educ.*, vol. 18, no. 6, pp. 633–637, 2002.

[8] J. D. Mccowan, "An integrated and comprehensive approach to engineering curricula, part two: Techniques," *Int. J. Eng. Educ.*, vol. 18, no. 6, pp. 638–643, 2002.

[9] J. D. Mccowan, "An integrated and comprehensive approach to engineering curricula, part three: Facilities and staffing," *Int. J. Eng. Educ.*, vol. 18, no. 6, pp. 644–651, 2002.

[10] J. D. Mccowan and D. R. Voltmer, "Curriculum for an engineering renaissance," *IEEE Trans. Educ.*, vol. 46, no. 4, pp. 452–455, Nov. 2003.

[11] S. C. Hu, "A wholesome ECE education," *IEEE Trans. Educ.*, vol. 46, no. 4, pp. 44–451, Nov. 2003.

[12] A. L. Sangiovanni-Vincentelli and A. Pinto, "An overview of embedded system design education at berkeley," *Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 472–499, 2005.

[13] K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "An embedded systems curriculum based on the IEEE/ACM model curriculum," *IEEE Trans. Educ.*, vol. 51, no. 2, pp. 262–270, May 2008.

[14] H. Rehman, R. A. Said, and Y. Al-assaf, "An integrated approach for strategic development of engineering curricula: Focus on students' design skills," *IEEE Trans. Educ.*, vol. 52, no. 4, pp. 470–481, Nov. 2008.

[15] A. Mohan, D. Merle, C. Jackson, J. Lannin, and S. S. Nair, "Professional skills in the engineering curriculum," *IEEE Trans. Educ.*, vol. 53, no. 4, pp. 562–571, Nov. 2010.

[16] B. Haetzer, G. Schley, R. S. Khaligh, and M. Radetzki, "Practical embedded systems engineering syllabus for graduate students with multidisciplinary backgrounds," in *Proc. 6th Workshop Embed. Syst. Educ.*, 2011, pp. 1–8.

[17] B. Kienhuis, E. F. Deprettere, P. Wolf, and K. A. Vissers, "A methodology to designing programmable embedded systems—The Y-chart approach," in *Proc. SAMOS*, 2002, Lecture Notes In Computer Science, pp. 18–37.

[18] D. Gajski, *Silicon Compilers*. Reading, MA, USA: Addison-Wesley, 1987.

[19] M. Rashid, I. A. Tasadduq, Y. I. Zia, M. Al-Turkistany, and S. Rashid, "A methodology for the assessment of pedagogic and implementation aspects of laboratories," in *Proc. FECS*, 2012 [Online]. Available: http://world-comp.org/p2012/FEC2650.pdf

[20] M. Rashid, I. A. Tassaduq, Y. I. Zia, M. Al-Turkistany, and S. Rashid, "Evaluation of engineering laboratories," in *Proc. ICEELI*, Jul. 2012, pp. 1–6.

[21] ABET Engineering Accreditation Commission, Baltimore, MD, USA, "Criterion for accrediting engineering programs," 2012.

[22] A. Mohan, D. Merle, C. Jackson, J. Lannin, and S. S. Nair, "From student to teacher: Transforming industry sponsored student projects into relevant, engaging, and practical curricular materials," in *Proc. Transform. Eng. Educ., Creating Interdiscipl. Skills Complex Global Environ.*, 2010, pp. 1–10.

[23] International Technology Roadmap for Semiconductors, "2012 update overview," 2012.

[24] R. Bergamaschi, L. Benini, K. Flautner, W. Krujtzer, A. Sangiovanni-Vincentelli, and K. Wakabayashi, "The state of ESL design," *IEEE Design Test Comput.*, vol. 25, no. 6, pp. 510–519, Nov.–Dec. 2008.

[25] N. Bombieri, F. Fummi, and G. Pravadelli, "Automatic abstraction of RTL IPs into equivalent TLM descriptions," *IEEE Trans. Comput.*, vol. 60, no. 12, pp. 1730–1743, Dec. 2011.

[26] H. Scharwachter, D. Kammler, R. Leupers, G. Ascheid, and H. Meyr, "A retargetable framework for compiler/architecture co-development," *Design Autom. Embed. Syst.*, vol. 15, no. 4, pp. 311–342, 2011.

[27] A. Gamatié, S. Le Beux, E. Piel, R. Ben Atitallah, A. Etien, P. Marquet, and J.-L. Dekeyser, "A model-driven design framework for massively parallel embedded systems," *Trans. Embed. Comput. Syst.*, vol. 10, no. 4, pp. 39:1–39:36, 2011.

[28] M. H. Imam and I. A. Tasadduq, "Evaluating the satisfaction of ABET student outcomes from course learning outcomes through a software implementation," *Int. J. Quality Assurance Eng. Technol. Educ.*, vol. 2, no. 3, pp. 21–33, 2012.

Muhammad Rashid received the Bachelor's degree in electrical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2000, the Master's degree in embedded systems design from the University of Nice, Sophia-Antipolis, France, in 2006, and the Ph.D. degree in embedded systems design from the University of Bretagne Occidentale, Brest, France, in 2009.

He is an Assistant Professor with the Computer Engineering Department, Umm Al-Qura University, Mecca, Saudi Arabia. Prior to joining Umm Al-Qura University, he worked with Thomson Research and Development, Paris, France, and Advanced Engineering Research Organization, Wah Cantt, Pakistan. His research interests mainly include electronic design automation for embedded systems and engineering education.

Imran A. Tasadduq received the Bachelor's degree in electrical engineering from N.E.D. University, Karachi, Pakistan, in 1990, the Master's degree in systems engineering from King Fahd University, Dhahran, Saudi Arabia, in 1994, and the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2002.

He is a Professor with the Computer Engineering Department, Umm Al-Qura University, Mecca, Saudi Arabia. Prior to joining Umm Al-Qura University, he worked with the University of Ottawa, Ottawa, ON, Canada; King Fahd University; Sir Syed University, Karachi, Pakistan; and FAST-National University, Islamabad, Pakistan. His research interests are in multidisciplinary optimization, engineering education, and various domains of wireless communications such as OFDM and MC-CDMA.