A Method for Evaluating a Computing Program's Continuous Improvement Plan

Abdullah Azzouni
Computer Science Department
Oregon State University
Corvallis, OR, USA
azzounia@oregonstate.edu

Jennifer Parham-Mocello
Computer Science Department
Oregon State University
Corvallis, OR, USA
parhammj@oregonstate.edu

***Abstract*—In today's competitive academic environment, academic computing programs must continuously improve, and for accredited programs, establishing and documenting a continuous improvement (CI) plan is a main requirement for accreditation. While many academic computing programs strive to implement a comprehensive CI plan that addresses all angles of the process, which we call 360-CI, they rarely do. One of the reasons for this deficiency is the ambiguity of what comprehensive CI (or 360-CI) is. From the literature, we identify 8 components of CI that should be addressed in every academic computing program's CI plan. These components include Administration, Curriculum, Course, Faculty, Research, Academic Advising, Facilities, and Support Staff. Each CI component is not addressed equally in the literature. The most emphasis is on Curriculum, Course, and Faculty, while the others receive much less attention.**

In this paper, we introduce an "ideal" 360-CI model utilizing all 8 CI components, and we use the 360-CI model to develop a method for scoring the comprehensiveness of an academic computing program's CI plan. To evaluate this method, we conducted a series of 21 semi-structured interviews and follow-up questionnaires with administrators and faculty in a large electrical engineering and computer science program. The results are consistent with the literature showing the most emphasis on Curriculum, Course, and Faculty CI and the least emphasis on Advising, Facilities, and Support Staff CI. Based on the results from this research, we propose potential approaches to help academic computing programs establish and maintain a CI plan that maximizes their CI score.

***Index Terms*—Leadership, Faculty development, Accreditation, Organizational assessment, Multilevel program assessment, Interviews, Qualitative, Doctoral Institution**

I. Introduction

In today's fast paced advancement in computing, academic computing programs cannot afford to stop improving. Many of these programs implement continuous improvement (CI) plans to help in this effort. While many of those programs claim to have succeeded in establishing CI plans [1]–[3], there is no way of measuring the level of comprehensiveness of these CI initiatives.

A truly comprehensive model of CI addresses all angles of such a program with a 360° view (henceforth, 360-CI). These angles consist of the following 8 CI components: cUrriculum (U), Course (C), Administration (A), Faculty (F), Research (R), aDvising (D), faciliTies (T), and Support Staff (S) [4]. In documented practice and in research literature, these CI components have been emphasized in academic computing CI efforts to varying degrees. The most attention has consistently been paid to Curriculum (U), Course (C), and Faculty (F). All CI components can potentially be addressed within an academic computing program independently of each other, but this will still not produce a comprehensive CI plan.

Integrating the eight CI components is a key aspect of establishing a comprehensive plan. In practice, integrating all CI components with each other has not been reported. The maximum documented number of CI components in a single CI plan was 6 [5]. Even in the cases where different CI components were integrated, Curriculum, Course, Faculty, and Administration have a relatively high integration rate, while other components such as Facilities and Advising were almost never integrated.

In this paper, we briefly introduce the idea of an ideal 360-CI model. We then introduce a method to help quantify the comprehensiveness of a CI plan and discuss the results of applying this method to a large Electrical Engineering and Computer Science (EECS) program. Next, we propose an Academic Computing CI Scoring Survey (ACCISS and pronounced as access) to help more practically generate the CI score for academic computing programs. Finally, we list limitations of this research work and ideas for future research opportunities based on our findings.

II. Background

Establishing a successful 360-CI plan requires an academic computing program involve and integrate all 8 CI components. At a minimum, every CI component must conduct its own CI. In an ideal world, every single CI component must interact with every other CI component generating and utilizing data that helps in the overall CI of the program. This is represented in Fig. 1 where lines between different CI components are bi-directional meaning that each component collects data to help improve the other component. Arrows are omitted to reduce the complexity of the diagram.

This model helps quantify the comprehensiveness of an academic program's CI efforts by using the scoring scheme presented in Fig. 2. Each direction is worth 5 points, while a CI component's own CI loop is worth 10 points. The reason a CI component's own CI loop is a higher value is because

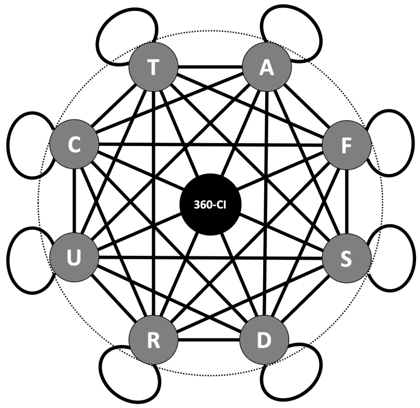

Fig. 1. Ideal 360-CI Integration

	C	T	A	F	S	D	R	U	
C	10	5	5	5	5	5	5	5	45
T	5	10	5	5	5	5	5	5	45
A	5	5	10	5	5	5	5	5	45
F	5	5	5	10	5	5	5	5	45
S	5	5	5	5	10	5	5	5	45
D	5	5	5	5	5	10	5	5	45
R	5	5	5	5	5	5	10	5	45
U	5	5	5	5	5	5	5	10	45
	45	45	45	45	45	45	45	45	360

Fig. 2. Ideal 360-CI Scoring Scheme

TABLE I
NUMBER OF IDENTIFIED CI DATA TYPES

Data Exch	Description	Total	EECS
U	Curriculum CI Data	9	4
U→F	Curriculum CI Data → Faculty CI	3	2
U→C	Curriculum CI Data → Course CI	3	3
U→A	Curr. CI Data → Admin. CI	1	1
U→D	Curriculum CI Data → Advising CI	1	1
F	Faculty CI Data	7	6
F→U	Faculty CI Data → Curriculum CI	1	1
F→C	Faculty CI Data → Course CI	3	3
F→A	Faculty CI Data → Administration CI	2	1
D	Advising CI Data	2	0
R	Research CI Data	3	3
R→F	Research CI Data → Faculty CI	1	1
R→A	Research CI Data → Admin. CI	1	1
R→T	Research CI Data → Facilities CI	1	1
C	Course CI Data	14	11
C→U	Course CI Data → Curriculum CI	10	8
C→F	Course CI Data → Faculty CI	10	7
C→T	Course CI Data → Facilities CI	3	0
C→D	Course CI Data → Advising CI	1	0
C→A	Course CI Data → Administration CI	1	1
A	Administration CI Data	8	3
A→U	Admin. CI Data → Curr. CI	3	2
A→F	Administration CI Data → Faculty CI	4	4
A→C	Administration CI Data → Course CI	4	4
A→S	Admin. CI Data → Support Staff CI	1	1
T	Facilities CI Data	1	0
T→F	Facilities CI Data → Faculty CI	1	1
T→C	Facilities CI Data → Course CI	1	1
S	Support Staff CI Data	2	0
S→C	Support Staff CI Data → Course CI	1	1
S→A	Support Staff CI Data → Admin. CI	1	1

	C	T	A	F	S	D	R	U	
C	10	5	5	5		5		5	35
T	5	10		5					20
A	5		10	5	5			5	30
F	5		5	10				5	25
S	5		5		10				20
D						10			10
R		5	5	5			10		25
U	5		5	5		5		10	30
	35	20	35	35	15	20	10	25	195

Fig. 3. Attainable 360-CI Scoring Scheme

a CI component should at a minimum collect data for the self-improvement feedback loop.

The maximum score of 45 for each row indicates how much the CI component on that row contributes to other components. Similarly, the score of each column (maximum of 45 as well) indicates the degree to which other CI components contribute to the column's CI component. The maximum total score in an academic computing program with a truly comprehensive CI will be 360 (another reason to call the model 360-CI). It is important to emphasize that what constitutes a score of 5 for an edge will depend on the unique circumstances of each academic computing program.

One has to keep in mind that in order to actually achieve a score of 360, an academic computing program must bidirectionally exchange types of data between all eight components, as well as collect types of data for itself. However, only a subset of types of data have been identified [5]. Table I lists a high-level summary of those identified data types. Based on this table, the maximum score for a program's CI plan can be 195, as shown in Fig. 3.

III. METHODOLOGY

In order to calculate the 360-CI score of the participating EECS program, a series of 21 semi-structured interviews were conducted with different stakeholders, such as faculty, instructors, and curriculum committee members in the program. The interview questions were open-ended to provide interviewees the opportunity to elaborate as much as needed. The questions covered the overall process based on the current understanding of existing continuous improvement research [6], [7]. We recorded the interviews, and the research team coded the transcripts after achieving 80% inter-rater reliability (IRR) on 20% of the collected data. The coding scheme consisted of multiple dimensions: attitude or sentiment, type of CI artifact, and context. A statement's sentiment was either positive, negative, or neutral. The CI artifact was either task, data, tool, or awareness. The context was either general CI or accreditation-specific. In total, there was 1,327 coded items (although many overlapped between interviews).

IV. Results

The types of data exchanged between components empirically identified in the participating EECS program contains a subset of all the attainable data exchanges in table I (see Table II for the CI data types and exchanges identified). In order to illustrate how to use this result to calculate the comprehensive CI score for this EECS program, we assume that an identified exchange of data between CI components in the interviews means that the data type is actually generated and utilized. This assumption is made for illustration purposes only.

To generate each cell in the scoring table, we calculate the ratio of data types identified in the interviews to the complete list of attainable CI data exchanges identified from the literature and interviews (see columns 3 and 4 in Table I). For example, we identified 14 different types of course CI data generated and utilized for the purposes of improving courses, and the interviews identified 11 of those possible data types. Thus, the score for this EECS program's Course (C) CI is 11 out of 14, but because we put more emphasis on collecting data for the purposes of self-improving a CI component, we weigh it more than exchanges of data from one component used to improve another component. For instance, the EECS program's Administration (A) CI collected and exchanged 2 out of the 3 data types identified for Curriculum (U) CI (A→U), but this is weighed less. Below, we present examples of these equations used for scoring exchanges within and between components.

$$10_{(max\ score\ for\ own\ component)} \cdot \frac{11_{(program\ data\ types)}}{14_{(total\ data\ types)}} \approx 7.9$$

$$5_{(max\ score\ for\ crosss-component)} \cdot \frac{2_{(A \rightarrow U\ data\ types)}}{3_{(total\ A \rightarrow U\ data\ types)}} \approx 3.3$$

Using these equations and the ideal scoring matrix presented in Fig. 2, we score this EECS program as a 131 out of an attainable score of 195 (see Fig. 4). This score on its own gives a high-level indication of how comprehensive an academic computing program's CI is. The closer this number to a "full-mark" or 195 on each axis, the more comprehensive the program is. However, the score is not as important at looking individual cells, rows, and columns where the score might indicate areas of satisfactory CI processes and, more importantly, areas where improvement is needed. If we take the scores for course CI, for example, it has a local score of 7.9 out of 10. That could indicate a substantial amount of CI within that component. At the same time, the course CI **row** score is 20 out of 35 (on the attainable-CI scale) for contributing to other CI components. Whereas, the Course CI **column** score is 33 for the contribution it receives from other CI components. This indicates that this EECS program is not using the outcomes of course CI as much in improving other components while other components contribute significantly more to course CI. Again, this example mainly aims to explain the concept of the 360-CI scoring scheme. As the scoring scheme and list of data types and exchanges grow, then the best attainable score will move closer to 360.

TABLE II
Examples of CI Data Types Identified at the EECS Program

U – Curriculum:
- U02: Employer Surveys
- U08: Course Objectives
- U09: Program Recommendations
- U10: Curriculum Details

UF – Curriculum → Faculty:
- U08: Course Objectives
- U13: Teacher Knowledge & Skills

UC – Curriculum → Course:
- U08: Course Objectives
- U09: Program Recommendations
- U12: Required Course Changes

UA – Curr. → Admin.:
- U09: Program Recommendations

UD – Curriculum → Advising:
- U11: Ground Rules for Advising

C – Course:
- C02: Faculty Feedback
- C08: Course Evaluation Results

C – Course: (Cont.)
- C10: Student Results
- C11: Student Feedback
- C12: Assessment Fit to Objectives
- C19: Course Dependencies
- C20: Course Details & Syllabus
- C21: Student Demographics
- C22: Intangible Student Outcomes
- C23: Class Size
- C27: Similar Course Review

CU – Course → Curriculum:
- C01: Course Adequacy
- C02: Faculty Feedback
- C08: Course Evaluation Results
- C10: Student Results
- C11: Student Feedback
- C13: Student Outcomes to Objectives
- C19: Course Dependencies
- C20: Course Details & Syllabus

	C	T	A	F	S	D	R	U	
C	7.9		5	3.5		0		4	20
T	5	0		5					10
A	5		3.8	5	5			3.3	22
F	5		2.5	8.6				5	21
S	5		5		0				10
D						0			0
R		5	5	5			10		25
U	5		5	3.3		5		4.4	23
	33	5	26	30	5	5	10	17	131.2771429
	131.2771429								

Fig. 4. Participating EECS Program CI Score

V. Future Work

Conducting interviews and coding the data in order to calculate an academic computing program's 360-CI score is a lengthy, error-prone, and costly process. Having shown how the scoring scheme could be used, we propose a tool called ACCISS (pronounced like "access" and stands for Academic Computing Continuous Improvement Scoring Survey). This tool is a survey that can be administered program-wide or university-wide. ACCISS allows a program or a university to collect the information needed to determine their attainable comprehensive CI score using Fig. 3 and a modified Table I.

ACCISS will solicit information on all edges of the model to identify gaps and potential issues, as well as the score. To do that, every participant is presented with a set of questions about CI components related to their role. So, a faculty member will be asked about faculty CI and how faculty CI interacts with the other seven CI components. A course designer would be asked about course CI and how course CI interacts with other the other seven CI components. People with multiple roles (a faculty member who is a course designer for example) can be asked about the faculty and course CI components.

The survey results will help score an academic computing program's CI efforts against the attainable 360-CI model in Table I, as well as add additional interactions to the attainable model. For example, a survey question can be as simple as: "Does your program collect data for ___ component?" If the participant selects "Yes" for an intra-CI-component data type (data collected for its own CI), it can be scored as a 10. If the participant selects that they "Collect it on their own", then they get a 5, and if they are "Not Sure", then they get a 1. They get 0 if they say they do not collect any data. A similar approach can be used for inter-CI-component data types. An answer of "Yes" is scored as 5, while an answer of "No" is a 0 score. "Not Sure" can be a 1, while "Collect on My Own" is a 2. As additional data types are discovered, the table of data types in Table I and scoring scheme in Figure 3 should be modified to contain a new listing of all data types and new attainable score, as well as updating ACCISS.

The 360-CI model provides potential for many future research projects. The results of implementing ACCISS can improve the set of CI data types and their list of possible questions. Thus, an implementation and distribution of the improved ACCISS tool in different computing programs can lead to a higher attainable best possible score to encourage better 360-CI plans and processes.

VI. Limitations

The models presented here, the ideal 360-CI model (see Fig. 1) and the attainable 360-CI scoring (see Fig. 3), depend on the currently known set of CI components and component data types. While the current number of CI components is eight, new CI components can be identified in the future. This is also true for the the number of attainable data types and exchanges identified in the future.

The calculation method of the EECS program score is not meant to be accurate. This example serves as an illustration for how to apply the scoring scheme and assess a program's 360-CI. To produce a more realistic score, each academic computing program needs to generate a set of ACCISS questions corresponding to the list of known CI data types, along with additional questions to gather more data types and exchanges. Generating such a set of questions is not a trivial task but it can contribute to the body of literature of what true 360-CI is and how to attain it. Once ACCISS is established, further improvement to the model will be much easier.

VII. Conclusion

This paper summarizes previous work identifying 8 CI components required to establish a comprehensive CI plan or initiative in an academic computing program. A list of all known possible data types identified in the literature and empirically is also summarized. From this list, we propose an attainable 360-CI model and scoring scheme that can be used to find the comprehensiveness of a program's CI effort out of 195, rather than the ideal score of 360. We apply this attainable scoring scheme to a large EECS program using rough calculations based on a set of 21 interviews with a set of key stakeholders, and the EECS program receives a CI score of 131 out of 195. In addition, we discuss how a comprehensive CI score can help academic computing programs address areas of improvement.

In the last section, we propose ACCISS, a practical method to produce the data required to calculate the attainable 360-CI score. ACCISS is based on all the currently known CI component data types, where each data type receives one or more questions in the survey. The answers are then assessed to produce the final 360-CI score.

A more tangible outcome of this work is its ability to help academic computing programs pinpoint areas and CI components that need improvement the most. This work also helps those programs devise holistic and comprehensive CI plans, rather than the disjointed and uncoordinated efforts that take place currently. Using the 360-CI model will help with reducing redundant tasks and optimizing resources. Also, developers of CI software tools can use this work to design applications, and computing programs can research the comprehensiveness of CI plans based on this work.

References

[1] Y. Al-Alawi, D. Al-Kaabi, S. Rashdan, and L. Al-Khaleefa, "Quality assurance and continuous improvement: A case study of the university of bahrain," vol. 15, no. 1, pp. 61–69, publisher: Routledge _eprint: https://doi.org/10.1080/13538320902731575. [Online]. Available: https://doi.org/10.1080/13538320902731575

[2] K. A. Buragga and M. A. Khan, "Successful ABET accreditation at king faisal university – rubric based assessment plan for continuous improvement."

[3] C. Wei, L. Feng, Z. Shu-ren, H. Shi-ming, and Z. Ying, "A research on continuous improvement of computer science and technology specialty based on engineering education accreditation." [Online]. Available: http://en.cnki.com.cn/Article_en/CJFDTotal-GJYJ201602020.htm

[4] A. Azzouni and J. Parham-Mocello, "Continuous improvement in academic computing programs is rarely comprehensive."

[5] ——, "Evaluating the current continuous improvement approach in abet-accredited computing programs," pp. 443–449, 2021.

[6] G. Anand, P. T. Ward, M. V. Tatikonda, and D. A. Schilling, "Dynamic capabilities through continuous improvement infrastructure," vol. 27, no. 6, pp. 444–461, publisher: Elsevier.

[7] J. W. Prados, G. D. Peterson, and L. R. Lattuca, "Quality assurance of engineering education through accreditation: The impact of engineering criteria 2000 and its global influence," vol. 94, no. 1, pp. 165–184, publisher: Wiley Online Library.