THE FOUR FORCES THAT INFLUENCES THE QUALITY OF KNOWLEDGE PROVISION SERVICES

Benjamin Hsueh-Yung Koo

Tsinghua University, Beijing, China

John Jianzhong Cha

Beijing Jiaotong University, Beijing, China

Edward Crawley

Massachusetts Institute of Technology

ABSTRACT

In this article, we present an engineering knowledge management methodology that aims to improve the quality of services of educational institutions. We first define educational institutions as agencies in a service industry that provide knowledge accumulation and dissemination services to their surrounding society. We then leverage the conceptual framework originally proposed by a Constitutional Law Professor, Lawrence Lessig, in the book “Code and Other Laws of Cyberspace”, to control the quality of services on knowledge management. We found his conceptual model of the four forces particularly suitable for understanding the complex interactions between technical content knowledge and the motivational forces in human societies. Based on this four-force model, we will use the design and implementation of a series of engineering courses at Tsinghua University to illustrate the concept.

KEYWORDS

Service Agency, Quality of Service, Architecture, Law.

INTRODUCTION

Engineering education is not only about improving the general human qualities of its target students, but also about offering knowledge provisioning services to the society. For instance, an engineering school should act as an agency that disseminates technical knowledge and inspires both current and future engineers’ ability to Conceive, Design, Implement, and Operate (CDIO, see http://www.cdio.org) modern technical artifacts. The objectives of an engineering curriculum should not only focus on individual performance, but also the collective service qualities of the entire institution. A good curriculum design must take into account the various socio-technical factors that are similar to designing a miniature society. Intellectually speaking, designing a curriculum for an engineering department or an engineering school, is analogous to writing a constitution. However, most engineering

curriculums are designed by professors or administrators that were primarily educated as engineering specialists. Engineering curriculum has usually been focused on students' individual understanding of certain technical subjects, with less emphasis on students' ability to systematically utilize the technical resources in their surrounding environment. Inspired by the four-forces model presented in Lessig's book on "Code and Other Laws in Cyberspace", we found that the law professor's mental model could enhance our understanding in the process of forming and executing an engineering curriculum.

What are the four forces?

Lessig defines the four-force model as the intrinsic motivating forces of social behaviours. His model provides a vocabulary to describe the behaviour of a school as an agency that provides services in its surrounding social and economic context. In a way, it provides a social scientist's framework to study the first CDIO standard, "Context". It could help curriculum designers to see the motivating forces and inherent constraints through a strategic lens. Lessig's four forces are: architecture, law, market and norm. The following definitions are the authors' interpretation of Lessig's forces, in the context of engineering education.

Architecture

Architecture is the kind of constraint that impacts a system before any activities take place. For example, a password protected information system would require its users to first obtain certain authorization, or expense a certain amount of intellectual effort to hack the security system, before one may tap into the protected information resources. In short, architecture is a kind of pre-emptive force that guides the behaviour of agents. This force is particularly relevant to engineering education, because engineering disciplines are naturally associated with technical artifacts. The workspace of an engineering school, the technical instruments available for teaching and research, and the organizational architecture of the school, can all be considered as the "architectural" force. Due to its pre-emptive nature, "Architecture" is a strategic item that should be planned to last a significant period of time. It is the stable element in a system [Koo 2005]. In contrast to "Architecture", the other forces, such as "Law", "Market", and "Norm", might be applied to constrain the collective behaviour based on case-by-case scenarios.

Law

Law is a set of explicit regulatory rules that could be executed after certain behaviour takes place. For example, a school may set up a "law" or "policy", that punishes students who frequently skip class. However, the effect of the "law" must be tested at two time points after it is established. First, it must wait until certain students actually miss enough number of classes, then, the other students must observe whether the punishment is properly executed with sufficient strength. In other words, the persuasive power of "Law" can be demonstrated only after certain activities takes place. If the award or punishment is too weak, it will not sufficiently affect the behaviour on a societal level. Moreover, its power primarily lies in the ability to execute the law in a consistent fashion. If the cost of law enforcement is too high, or it misses too many offensive acts, the law would have very little value. Its regulatory effect could only be as good as the agencies' (in our case, the engineering department or schools') ability to execute their "laws". Clearly, regulations or policies of a school should be planned along with the architectural elements of the curriculum. However, from a temporal viewpoint, "Law" is complementary to "Architecture", because its successful enforcement of the constraints happens after certain specified activities take place.

Market

Market is the force that pulls, or attracts the agents to participate in or react to certain self-interested opportunities. For example, media coverage about certain students' or teachers' achievements could be counted as a kind of market force. Similarly, students who might be given an award, or a ranking, or a higher mark for their learning effort could also be categorized as a force in the "Market". According to Lessig, the force of "Market" pulls agents to follow a certain direction. Clearly, if the market awards are not attractive enough, students or teachers could sneer at them. In short, Market is just an invitation; it relies on the agents' willingness to engage with the force.

Norm

Norm is a force that pushes, or actively encourages the agents to participate in certain activities. One may think of it as the force of culture, or peer pressure. When everyone in a school uses a network-based version control system to manage shared files, it would be a strong force to change the habit for people who only use email to send files. Clearly, there will always be people who run against the norms, but the forces in the culture would actively push, or more politely, encourage these people to act more "normally". Norm is a force that is much stronger than an invitation, it will try to push the "abnormal" agents into the mainstream, it doesn't rely on agent's self-willingness to engage with the movement.

A high level process for curriculum design

When applying the four forces to the design of curricula, we argue that the sequence of curriculum design activities should roughly follow the four stages. Architectures provide a stable framework in systems; therefore, they should be selected in the beginning. If the architectural choices are good, they should be leveraged for a long time, otherwise, they should be replaced as early as possible, because all the remaining forces will act in the context framed by chosen architectures. Law follows and complements architectural decisions, because it needs adequate administrative organization structures and the supporting technical infrastructure to execute the law. Market forces supplement what laws cannot cover, they should draw people, resources, and new ideas into the learning environment. Norms are shaped over time by slowly aggregating the effects of the previous three forces. Once a norm is established, it would serve as a filter to identify or reject micro-level behaviours that are not compatible with the collective whole.

First, we need to be cognizant of the consequences of architectural choices. For example, do we want to choose an organizational architecture that is flat, where students can freely interact with each other without an explicit team structure? Would this organizational architecture best support our learning objectives? From an operational viewpoint, architectural decisions like this would have profound impact on the learning outcome, and they must be determined before classes start.

Second, one need to conceive and implement certain regulatory mechanisms to enforce the architectural intent. This might include grading policies, attendance policies, policies on class participation and policies against plagiarism. All these rules must be explicitly stated and effectively executed in ways that would improve student learning while not stifling the freedom and creative energies in the student community.

Third, students must be motivated by sufficiently attractive incentive systems. This is called the force of market in Lessig’s term. Normally, grades and ranking can be treated as effective currencies in school. There are also other kinds of intellectual currencies. For instance, certain marketable skill sets are attractive to students. In courses related to computing science, skills in popular programming languages and modelling tools are considered to be valuable, because students think that these skills could help them find jobs or solve problems in less time. These kind of knowledge assets can be utilized as market incentives.

Fourth, all human organizations have some cultural norms. It could be leveraged to promote effective learning. Ideally, one would create a positive cultural norm, so that students and teachers will automatically encourage each other to raise their quality of work, be more creative, or just encourage each other emotionally when something didn't work as expected.

A CASE STUDY AT TSINGHUA UNIVERSITY

For illustrative purposes, we use a series of required courses in the Industrial Engineering Department of Tsinghua University, to show how we designed a CDIO curriculum based on the four-force model. Specifically, we want to show that by treating a class of students as a miniature society, four forces could naturally fit into the planning and execution of learning activities.

The two courses, "Data Structures and Algorithms", and "Database Systems", are chosen for detailed explanation in this study, partially due to their close relationships. One is the theoretical foundation of the other, and our department conducts these two required courses in two consecutive semesters. Administratively, this matches CDIO standard 3, "Integrative Curriculum". In terms of supporting infrastructures, both courses could leverage massive amounts of digitized content and tools that are freely available on the Internet. Over time, this makes it particularly easy to compare student performance given different technologies.

Architectures and functional goals

Architectural decisions include three aspects: technical, organizational, and workflow. Baldwin et al.'s book on Design Rules [Baldwin 2000] inspired us to divide curriculum planning in this fashion. In this paper, technical architectures refer to the technical content of a course. Should one focus on describing relational databases as the main theme, or start the course with a detailed discussion of declarative programming language to establish the foundational concept of a course? These two drastically different approaches determine the "technical" architecture of the course, and the differences could impact how students learn ongoing courses differently. Organization architecture refers to how students and teachers are administratively organized. For example, students can be organized into groups, and teachers organized into departments and research centers. Workflow architecture specifies the interactive patterns among human activities, physical instruments, and information flows. Specifically, weekly classroom lectures, bi-weekly lab sessions, and team-based final reports are typical workflow patterns. These simple terms actually involve sophisticated human and material interactions.

The technical content architecture, the organizational architecture, and the workflow architectures, must be aligned with a set of flexible, yet instructive functional goals. Before we make any architectural decisions, we decided to invite students to work with teachers to design a collective learning system. To guide the thought process of this design challenge, we came up with a set of functional goals expressed in the following words:

1. Knowledge Aggregation
    a. Aggregate people and resources to better retain existing knowledge
    b. Aggregate experience and traditions as a technology conservatory
    c. Aggregate momentum to develop future technical opportunities
2. Knowledge Dissemination
    a. Disseminate knowledge to students
    b. Disseminate knowledge across different departments and schools
    c. Disseminate knowledge to surrounding societal participants
3. Knowledge Creation

a. Create knowledge based on incremental aggregation
b. Create knowledge based on cross-pollination of disciplines
c. Create knowledge based on innovation

First and foremost, this set of functional goals is not a set of compliance metrics. For example, administrators might use numerical measurements as the “goals” of certain educational program. Similarly, many students believe that their goals in school are getting a degree, and getting top scores in certain tests. To inform the students and teachers that learning is about acquiring knowledge, not about fulfilling some abstract metrics, we decided to write down these goals and use them as guidelines to help them design their own learning activities.

These functional goals spell-out the operational purpose of a generic learning organization, which we call “Knowledge City” or KC in short. Initially, KC is just a code name that refers to a project that integrates all other teams’ work in the database course. Later, we realized that its goals are aligned with the goals of our whole department. Overtime, these goal statements become a central part of our course material, because they provide hints to run group-based learning operations. They also serve as a source of inspiration to help students explain why their self-chosen projects in both data structures and database courses are valuable. Invariably, all interesting projects must leverage some knowledge and support from people outside of their group, even outside of the campus.

These goal statements are also helpful in making architectural decisions during curriculum design. When presenting technical content of a course, the overarching themes of the lectures should reflect these purposes. When organizing student teams, the motivating factor is not about beating the other teams, but about serving these collaborative goals. When designing learning workflows, the activities and interactions are checked against these goal statements. These functional goals are used in the beginning of the class, to help all stakeholders of the course to choose the appropriate technical, organizational, and workflow architectures of the courses.

The three architectural concerns

Learning activities in our courses must contribute to the goals of knowledge aggregation, dissemination, and creation. These activities are also pre-emptively constrained by certain pre-established structures, or "Architectures". In this paper, we present three architectural concerns, namely technical architectures, organizational architectures, and workflow architecture. The following sections explain their roles in the design of our two courses.

Technical Architecture

The main technical goal of data structures and database courses is to aggregate, disseminate, and create knowledge about how to manipulate, store, and present data, using computing devices. Following ideas presented in the famous textbook, “Structure and Interpretation of Computer Programs” [Abelson 1996], we found that the technical content about data structures and computer programs can be established on a rather simple principle: *how to use mathematical abstraction to approximate concrete phenomenon using computing devices*. To ensure students could immediately realize the concrete implication of these abstract mathematical concepts, the textbook uses Lisp, an interpretive programming language, to engage students with the concrete world of computation. The design of this textbook provides a technical architecture of similar courses. From a pedagogical viewpoint, one may conclude: learning activities should be designed to illustrate, challenge, and demonstrate the connections between the abstract mathematical concepts and the concrete applications of computing technologies. Lectures should be organized to sequentially introduce the conceptually inter-dependent techniques of mathematical abstraction, and

more specifically about various abstract data types (ADT), such as set, list, tree, and graph. Then ADTs become the officially sanctioned vocabulary to describe concepts and examples in class. A technical architecture for knowledge aggregation, dissemination, and creation is established on top of the elements of a scientific language. As students advance in class, they acquire more vocabulary in class, which expands their ability to express their technical ideas. For example, when they learn about sorting and searching, they will not only acquire the "verb" phrases in their technical language, they will also learn bout the measurement metrics, such as the computational time complexity, and memory-space consumption. When they learn about various programming languages or database technologies, their vocabulary also increases, so that they can relate their abstract mathematical terms with increasingly richer and more concrete technology applications. We summarize these ideas into one sentence:

The vocabulary, grammar, and the application context of a scientific language, determines the technical architecture of engineering courses.

Courses related to data structures, algorithms, and databases naturally involve programming assignments. To students of introductory data structure classes, this course might be their first exposure to the reality of software engineering. Often times, students consider programming assignments as tasks, not software engineering projects. In our courses, we want to use programming assignments as an opportunity to practice software engineering. To conduct software engineering projects, we deployed an SVN-based version control service [SVN website] on our campus network. This service provides the ability to synchronize different authors' documents, source code trees, and other digitized information assets on a central repository. This technology provides a mechanism to compare and record the changes between versions of documents and therefore emulates the engineering environment of an Internet-based software development community. There are various open source tools that can automate the process of analysing changes between versions of source code. This includes the authorship, frequencies of changes, amount of changes, and the correctness of changes. Version control technologies provide a technical platform can be used to inform grading decisions, and detect plagiarism. According to our functional goals, we primarily use it to help students witness each other's learning progress. Without these open source technologies and an existing network infrastructure, conducting engineering courses that require frequent interactions and collaborations would be much more challenging.

We use Moodle [see http://moodle.org], an Open Source course management websites, to provide students with blogs, calendaring, and file sharing services. Instructors also communicate with these students by posting homework assignments, and lecture notes on the website. We choose Moodle primarily because it is a mature Open Source product, and has a large community developing modules that support a wide range of esoteric functions. This technical infrastructure also helps us better observe students' learning activities on a fine-grained level. We can see who are most vocal on their blogs, and who constantly share information with others. For people who are less active, we may also schedule private meetings to learn about their progress. The adoption of these tools improves the community's awareness level of media literacy [see http://newmedialiteracies.org/].

Higher levels of technical literacy came from students' direct experience with the new technical infrastructures. Version control software gives them automatically generated "version numbers" and customizable "branch names" to manage the evolutionary history of their digital media. Moodle gives them online "team-based discussion boards" to post questions and share knowledge across different teams. These technical infrastructures expand the vocabulary in students' daily conversation. The expansion of their vocabulary doesn't only make them speak like (software) professionals; it also improves their efficacy and efficiency in communicating discipline-specific ideas.

Organizational Architecture

Organizational architecture is a constraining force that regulates how people interact with each other. Given the above-mentioned technical architecture, students perceive themselves as the citizens of Knowledge City, and a people that understand a common language. They can attain this social identity partially because the technical infrastructure and architectures can be used beyond just one course. In Knowledge City, instructors of each class are politicians, or Mayors, who always delegate tasks to Vice Mayors, or teaching assistants. Teaching assistants are inspectors, who observe students' group and individual learning habits. Their jobs are not just about answering technical questions, they use above-mentioned tools to judge students' habitual behaviour. They observe whether individuals and teams post messages and read frequently asked questions (FAQs) on Moodle. They examine whether students adequately upload and modify source code on the source-code repository. Students are organized into teams. Each team plays the role of a service agency or a product manufacturer. In the beginning of a semester, each team comes up with a name and a logo. Within the first two weeks of a semester, they need to submit and present a proposal on how their firm could serve their potential customers in Knowledge City. In most cases, their customers are other teams in the same class. Then, each team will self-organize their local administrative structures based on their respective proposals.

In project-based learning, teams are organized to develop products together. Products in our two courses include: team-based student lectures, progress reports, individual homework assignments, in-class quizzes, and team-based projects. Based on our experience, many students simply don't have the knowledge or experience that it requires iterative negotiation cycles to integrate individual work. Normally, student teams just come up with a list of tasks, and assume that once individual tasks are done; they can be easily assembled into a complete and functional product without expecting additional work. Based on this mindset, students would usually work alone, and then be stunned by the often-disastrous outcomes when things need to be put together.

To address this naive project-development mindset, the organizational architecture must make teams be cognizant of their surrounding context; all products and technical content delivery must answer the needs of some living and breathing customers. Students must repeatedly negotiate with the instructor and the TAs to refine their team projects. Under the organizational context of Knowledge City, each team must defend their technical merits and technical contribution to the city's infrastructure. All their oral and written technical reports must include sections that explain the social value of their projects in the City. They are also encouraged to integrate technologies contributed by other teams. The notion of a city helps students articulate various business and social interactions between teams. The projects are not about making one outstanding team, but about creating a harmonious society consist of inter-dependent chains of product development teams.

Size matters. In a class of 60 to 80 people, there are usually 10~12 teams. Each team is allowed to have 5~9 student members. In our department, most students stay in the same dormitory and take the same set of courses. This organizational background makes it easier to coordinate team schedules. Occasionally, there are students coming from other engineering departments. They are usually the minority group, so that they can often accommodate the majority's scheduling needs. Due to the intensive amount of workload in these two courses, instructors who conduct courses in the same semesters often are from our department, and they are very understanding of students' overall workload. With the political support from the department, these organizational features strengthen students' willingness to work on projects related to Knowledge City.

Workflow Architecture

Technical architecture determines what is to be learned. Organizational architecture determines who will be responsible for different types of learning activities. Workflow architecture determines how to learn. Using Knowledge City as a context, learning naturally takes place before and after classroom-based lecture hours. Overtime, students compiled a set of workflow patterns. They are listed as follows:

1. In-Classroom Workflow (Roughly 3 hours a week)
    a. Instructor Lecture
    b. Student review on content presented in the previous week
        i. Show a 3 minute video from the previous week
    c. Student lecture (a lecture using their team project as an application context)
    d. Randomly selected student team progress report
    e. In class quiz
2. Content-Preparation Workflow (Weekly cycle)
    a. Lecture Topic Definition (derived from the Syllabus)
        i. Intra-team discussion with support from previous team.
    b. Create the alpha version of Lecture Notes (story boarding)
    c. Rehearsal with instructors and teaching assistants
    d. Create the beta version of Lecture Notes (detailed content)
    e. Rehearse with instructors and teaching assistants
    f. Prepare a quiz based on student-created Lecture Notes
    g. Upload all digitized content onto KC's source code repository
3. Content-Refinement Workflow
    a. Review with instructors immediately after lecture hours
    b. Create a list of errata
    c. Create a review presentation, this includes
        i. Key concepts in this past lecture
        ii. Information that was missing or incorrect in the lecture
        iii. Additional references
        iv. A 3 minute video, highlighting inspiring moments in lecture hours
    d. Rehearse the review presentation in team
    e. Upload all digitized content onto KC's source code repository
4. Weekly Team-Project Report Workflow
    a. Compile a list of relevant content knowledge gained from weekly lectures
    b. Continue development of team-based products
    c. Prepare a progress report with technical demonstrations (5~9 slides)
    d. Upload progress reports and demos onto KC's source code repository

This list of workflows is not intended to be comprehensive. They provide constraining guidelines to regulate learning activities in different timescales. The list of workflows is documented to provide an architectural guideline to orchestrate learning activities. All courses in Knowledge City are run as multi-threaded programs. When one team is creating a presentation, the other teams could have been developing technologies that might serve as an example in the presentation. If the presentation didn't have a chance to include it in an earlier week, they are encouraged to include them in later weeks. Students are also encouraged to include content derived from other courses. Classroom-based lecture hours are considered to be weekly synchronization points. Like townhall meetings, they can only highlight learning opportunities, but opportunities must be captured after meetings.

To evenly distribute the workload, each team is responsible for organizing at least one week's worth of learning activities. If a team fails to meet the minimum standards, they are required to do it again. Each week's improvements are used as the minimum standard for the next week. To deliver accumulative quality improvements, teams must work with previous

teams to transfer experience. These requirements naturally trigger intensive learning activities after lecture hours. The increasingly challenging standards make following teams want to start their preparation work early. Based on student feedback, some students started preparing for the course a year ahead of time. This workflow architecture emphasizes incremental improvements and iterations. The synchronization points in workflows help students better manage their time. This workflow architecture also de-emphasizes test scores and student ranking. In this workflow, tests and scores are employed as tools to diagnose and improve learning, not as the goals of learning.

Laws that complement architectural elements

Architectural elements only establish pre-emptive constraints, laws specify the consequences of constraint violation. For instance, Internet-based source code control systems provide a technical infrastructure to monitor by whom and when assignments are submitted. Students must obtain user accounts and passwords before they can participate in this networked community. However, this technical architecture provides no mechanisms to ensure that students would actually commit their assignment on time. The motivation of submitting assignments on time; comes from “laws”, such as grading policies in the courses. The effectiveness of laws is directly associated with the community’s ability to enforce the execution of these laws.

Law execution consumes energy, time, and labour. Ideally, good architectural elements would not only reduce violation of laws, but also make it easy to execute the laws. Higher levels of technical infrastructure support provide the means to execute relevant laws. For example, version control software has features to automate the process of detecting late submission. It can also perform source code differencing to identify plagiarism. Laws, or grading policies, should be designed after the established architectural elements, and executed under the support of good technical, organizational, and workflow architectures.

The power of law should be derived from the power of reasoning. People need to be consciously aware of their own crime when they violate laws. Otherwise, when people unconsciously violate the laws, they tend to consider whatever legal consequences to be surprisingly harsh. When legal consequences induce emotional reaction, they will diminish the power of law to constrain human behaviour. In the case of plagiarism, many students simply don’t know how to properly cite the sources. Under innocence and ignorance, they might frequently commit “intellectual crimes” without knowing it. To address this problem in Knowledge City, we spell-out the differences between knowledge aggregation, dissemination and creation. Through the differentiations between different kinds of knowledge provisioning services, students learn to become aware of legal versus illegal intellectual practices.

In the process of knowledge aggregation, students gain intellectual credits by demonstrating their scholarship in the historical origins of certain content knowledge. In both data structure and database courses, students need to know the inventors of certain algorithms, when they are invented, and what tools use these algorithms for what purposes. Laws play a role here by judging the precision of their scholarship.

In the process of knowledge dissemination, teams are judged by their ability and compliance in making content accessible to others in various formats. First, they need to organize the technical content of interests in a presentable format, so that the recipients of this content knowledge can follow the messages being disseminated. Oftentimes, badly organized content prevents the audience from finding references to cited content, therefore compromising the integrity of the disseminated message. Moreover, students need to be aware of whether the act of knowledge dissemination is suitable. For example, in the database course, we discuss what kind of data is easily attainable and why? What kind of privacy concerns of demographic data stops us from sharing certain information? Are there

copyright laws that limit the distribution of certain data content? In this process, our goal is to improve the literacy of intellectual ownership and copyright laws to reduce unnecessary violations.

In the process of knowledge creation, students must go through a rigorous reasoning exercise to validate the claim of knowledge creation. Students need to present an argument that establishes the intellectual basis of knowledge creation. This exercise usually takes place when students start formulating their thesis in research papers. Many also quickly realize that it could be very convenient to work with others to systematically examine claims of knowledge creation. Knowledge creation requires teamwork, even if the discoverer is a single person. Claiming a creation is very similar to applying for a patent or going through procedures in court. To validate a creative idea involves many witnesses. It needs to have accumulated a relevant set of facts, and the facts must be presented to witnesses in a well-organized manner. Without a social context and the preparation in the two previously mentioned processes, creativity would appear lawless. As stated earlier, the power of law comes from the power of reasoning. Students need to learn to defend themselves and argue for their claims. These debate sessions usually take place in the instructor's office, because it is difficult to anticipate the time required to defend a creative idea. These learning activities help students to develop their litigation skills on engineering subjects. Students' awareness on how to claim intellectual contributions could systematically reduce the chance of unintentionally violating the code of conduct in Knowledge City. To prevent people from intentionally breaking the code of conduct, such as cheating or just being lazy, we need to analyze market incentives and cultural norms in the City.

Markets and the currencies of knowledge

In a university environment, students often want to obtain higher grades to improve their grade point average. Instructors often care about publications and student feedback to support their tenure cases. Teaching assistants are usually working on research projects, therefore, they often need new ideas and volunteers to help them conduct experiments. In Knowledge City, we consider grades, student feedback (students grade the instructors), and time commitment as different kinds of currencies. Under different contexts, these currencies have different exchange rates between each other. This section explains how currencies are measured and used in the two courses.

Grade Assessment

For most students at Tsinghua, grades matter. They need high grade point averages to apply for graduate studies. They also need to be ranked in the top fifty percent to be considered for certain paid public service positions at school. Grades are a very concrete form of currency. It directly motivates students to follow grading policies as prescribed by instructors. In this sense, instructors and teaching assistants own the power to distribute this currency. Normally, instructors would prescribe the weighting factors for different kinds of assignments in the syllabus. Students often have little input to the syllabus.

In Knowledge City, a syllabus is like a business contract between teachers and the students. We engage former and future students to edit the syllabus and ask for feedback in the beginning of the semester. We want both teachers and students to think about whether the grading policies are acceptable to them. The process of involving students to refine the syllabus sends an important message to students. Grading policy is just a somewhat arbitrary mechanism; there is not a lot of science to it. They have the power to determine their own grades, including the mechanism to calculate their final grade. This process makes them think about the objectives of learning. It is not the grade that we are after, it is the knowledge acquired in the learning activities that has more value. We instruct the students to come up with grading policies that support the functional goals of Knowledge City. They soon

realized that it is not that relevant, because a grade is a form of conservative resource, like oil or gold. It can be freely distributed without exhaustion. If they work toward the goals, they can get a passing grade. To distinguish between their level of success, higher grades go to people who contribute more to the City. People should compete on how much they collaborate with others. The process of establishing this grading policy motivates students to focus on learning outcomes, not gaming the system to get a higher grade.

Student Feedback

In many schools, students fill out teacher evaluation forms at the end of a semester. At Tsinghua University, these evaluation forms may directly affect their cases for promotion. To please students and get high marks in return, teachers might soften their academic standards and give bloated high marks to students. This exchange mechanism doesn't necessarily motivate teachers to become better teachers. Contrary, it has the potential to compromise teachers' authority.

To improve Tsinghua's educational practice, the leaders of Industrial Engineering Department, specifically Gavriel Salvendy and Li Zheng (郑力), convinced the university level administration of Tsinghua, to conduct an experimental program for innovative teaching in the Spring semester of 2008 in the department. They chose three courses, and conducted a few experiments to test whether different lecturing styles have different effects on students' creativity. During this experimental period, the administration agreed to remove student feedback as a component in the corresponding teachers' promotion cases. This special policy applies to the Data Structures and Algorithm course, and two other related courses in the same semester. To compensate for the extra work caused by the experiment, all participating lecturers and teaching assistants received a reasonably attractive financial reward.

In Knowledge City, we consider student feedback a critical element to help us improve the learning experience. However, we don't want to turn students and teachers into bean counters. Badly designed reward systems often distort the original administrative intent. A school is a human society. It is willing and capable of change. When presented with reasonable arguments, the political system in the larger context may act favourable to support innovation. It is up to all members in the community to propose incremental changes to the larger context. Overtime, we believe that concepts originally experimented with in the Knowledge City will be able to help better design the linkage between student feedback and lecturers' incentive systems. For example, in Knowledge City, students are invited to help design their own grading scheme. Similarly, there are many other experiments we can conduct in the City to identify better currency exchanges between different stakeholders in the community.

Time Commitment

Time is a precious resource for most people. When it is expensed, it cannot be replenished. Learning is a time-consuming process; it takes even more time to think about, and to refine a system that would improve the effectiveness of learning. When Knowledge City was first conceived, processes, services, and infrastructures to improve learning must be designed, implemented, and operated from a blank slate. These resources and services require students, teaching assistants, and lecturers to devote their personal time to build something out of nothing. As most engineering projects, initial attempts may encounter setbacks, and it was never clear what would have been the material benefits of trying to do something innovative on the subject matter of "learning". We often joke that university administrative structures are the slowest changing components in human society. Based on this observation, it is not administratively obvious for an engineering department to expend its

resources in the highly abstract domain of helping people to learn. Conventionally, engineering professors and their students should focus on concrete applications of scientific disciplines, so that it may pragmatically and technically demonstrate values of the time expensed.

In different marketplaces, value judgements are different. The academic discipline of Industrial Engineering is to identify improvement opportunities on a system level. When you want to build one rocket, you turn to rocket engineers. When you want to build one thousand rockets, you need to apply the principles of industrial engineering. Similarly, to teach one student, one needs to find a good teacher. To teach one hundred students, many principles of industrial engineering could help. At the Industrial Engineering Department of Tsinghua, and the Engineering Systems Division at MIT, it is our technical interest to study the complex interactions between humans and technologies. In other words, spending time on the design of learning organizations can be easily justified within engineering departments that study systems. In our biased view, we believe that all other engineering department should also be interested in the complex interactions between human and technologies. However, each department will likely bring in a different angle to enrich the marketplace of useful solutions.

In Knowledge City, all students and teaching assistants are explicitly informed that their goal is to improve the performance of the learning community as a whole. We invite students to come up with mathematical abstractions to measure their contribution toward the effectiveness of learning. We require students participate in team-based presentations and team-based projects, so that they can share their experience on learning new concepts and learning new tools. When sharing becomes a common practice, supported by reliable technical infrastructures, people start to benefit from their own contributions toward the community. Students who contribute more to the learning community gain respect from others. Their contributions are recorded in the database, so they improve their chance of receiving higher grades. Teaching assistants gain from sharing by knowing who are the most talented students in class. These students could complement teaching assistants in answering questions from other students. Lecturers gain from sharing by focusing in-class lectures on more advanced intellectual ideas. More fundamental ideas that could be explained by students and teaching assistants should be covered by students' team-based lectures, or in recitations conducted by teaching assistants or team-based study groups. In short, in the marketplace of Knowledge City, one spends time to gain more time. In many cases, the more you spend, the more you get out of it.

Norms and the effects of peer pressure

Conformity in a community has the power to push its citizens to establish certain habitual tendencies. Norm or conformity establishes social binding, and it can be leveraged to promote effective learning. In Knowledge City, conformity is utilized in a format similar to fashion shows. The town-hall meetings, or in-class lectures, are their opportunities to demonstrate their latest fashionable products. Each student lecture must be at least as stylish and as innovative as the previous team. Otherwise, members will receive pressure within a team, partially because their performance will be reflected on the team-based grade. If a team or an individual performed poorly, they will feel embarrassed in their respective community. The degree of embarrassment is related to the social norms. What is the normal amount of innovation one must exhibit to be considered normal?

The right to be different

We are not interested in indoctrinating students into a normalized regiment to tackle problems. Engineers must solve problems by recombining seemingly incompatible concepts and techniques into some working solution. Ideally, we want students to exert peer pressure in ways that keep other students interested and committed to their teamwork. We don't want

to instil a culture that blindly considers some tools or some approaches are strictly better than the others. For example, students are allowed and sometimes encouraged to mix and match Java, Mathematica, C and C# to create solutions in their team projects. We suggest they learn and use one programming language, namely Java, partially due to the ease of deployment and its popularity. For novice programmers, they can leverage a wide range of tutorials, documentation, and source code samples to help them learn. This doesn't mean that they should only use Java to solve all their problems. When students work with a language like Java, or C++, they tend to treat problem solving as a software engineering problem. They need to construct many class diagrams and collect many source code libraries before typing a line of source code. This also forced them to spend a lot of time to learn the syntax and existing libraries of the programming languages. Once they spent too much time on source code, they forgot the purpose of engineering is to solve an applied problem, not about demonstrating their programming skills. Certain tools tend to bring along a culture or norm to solve problems that may not be adequate to all problems.

To counter this Java-centric culture, we also introduced Mathematica as a programming and presentation environment, which shows them how to rapidly construct computational experiments and solve problems in an interactive manner. Mathematica is an interpretive language, it comes with a large functional library and visual programming components. For quick and dirty tasks, it bypasses many stages of software engineering practices. We want students to learn this different style of programming, so that they can see the norms in this type of work. Norms can be useful, when they are applied in the adequate context.

A shared value across Knowledge City

According to Richard Stallman, the founder of the Free Software Foundation [5]: "free software is a matter of freedom, people should be able to use software in all the ways that are socially useful." This philosophy ignited a software engineering revolution that created GNU, more popularly known as Linux, the first "free" operating system. It established a social norm that made a significant amount of well-engineered technologies available for free use.

We believe that this value must be deeply embedded in Knowledge City. The social value: "sharing is good", is different from "attaining competitive edge" as often mentioned in popular MBA courses. We establish the norm in our courses by demonstrating that engineering methods and technical resources usually create more value by having a larger community share the same standard. From an operational viewpoint, the world has already benefited from the philosophy and social norms created by the Free Software Foundation. It made many technologies freely available to the masses, created many egalitarian technical communities, and most importantly, this simple philosophy generated many high quality but low cost products. Without the technical achievements and social norms initiated by Richard Stallman, the two courses, data structures and databases, would be supported by a much smaller set of tools, therefore diminishing the forces that we could utilize the support learning activities outlined in this paper.

DISCUSSION AND CONCLUSION

We now go backwards from the force of norm, toward the force of architecture. In the 21st century, we have witnessed that the social norm of sharing engineering knowledge has generated many innovative products in various markets. The markets that endorse the exchange of open-sourced products and services also changed the practice of law, because it created new ways to interpret intellectual ownership. The lowered cost of engineering products also changed the architectural elements that support the functions of knowledge aggregation, dissemination, and creation. We argue that community-based learning activities

are driven by these forces, and curriculum planners should not only focus on the discipline-specific technical content, but they should see a class of students, or an engineering department as a miniature society. Then, it is possible to create an emergent learning society that is much more effective than treating individual students as the knowledge container.

Based on Lessig's four forces, this article presented a sequence of strategic considerations for designing a system-based engineering curriculum. We argue that curriculum planning must start with a set of architectural intent statements, or functional goals. Then, curriculum planners can proceed to select the technical, organizational, and workflow architectures for the subject matter of interest. Once the architectures are decided, we can leverage these architectural features to execute laws. Only through effective execution of "laws" or "school policies for learning", can the force of laws enhance the learning experience of the whole community. Otherwise, "laws" are just slogans and words that have no constraining powers.

Architecture and Law are the two forces for regulating activities before and after they take place. Market and Norm are the two forces that pull and push the entire learning community to learn more, or to learn with more effective methods and tools. These four forces complement each other, and they provide a new framework to think about curriculum planning and execution. In the past, engineering professors with deep domain expertise often designed curricula in ways that provide content knowledge to self-motivated individuals, so that they may eventually qualify for their future engineering tasks. In our proposed curriculum, students enrolled in a series of courses should be organized as a miniature society. This miniature society serves as a realistic context to practice collaboration, negotiation, and other survival skills that students must have before their graduation. To ensure the miniature society will productively deliver useful engineering experience to students, we described how the four forces model is used in Knowledge City to ensure delivery. Although our experience is limited to just one engineering department in one university, we believe that there are many elements of our work can be shared with other departments and universities to reduce their learning curve.

Balance and Symmetry

We need to first recognize that two courses in a series do not yet constitute a complete curriculum. However, we believe that the principles and techniques described in this paper are "symmetrical" across many types and scales of human organizations. In other words, the effect of these forces will not change when it is applied to situations that are much more complex these what we described here. We need to give credits to Lessig again, by mentioning that these forces were originally conceived in the context of the entire Internet-aware society. Therefore, we have the confidence that these forces can be used to balance many human activities independent of scale and type.

The administrative structures of organizations are often designed to include checks and balances. When we think of mobilizing a community of people to provide learning opportunities to each other, we need a set of symmetrical constraints that will cover all possible directions of movements. Architecture and Law complement each other on the time axis. Market and Norm are mirror reflections, because market pulls and norm pushes. These forces drive the quality of knowledge provision services in a learning community. We need a succinct mental framework that covers all bases; so that we can rely on constraining devices or pre-conceived correction actions to instil order. Without utilizing the vocabulary provided by social scientists, engineering knowledge provision will remain as a form of art.

Acknowledgement

The authors would like to thank Mr. Alan Greenberg, the Head of Learning Solutions and Strategy, Apple Computers, Asia. His constant infusion of ideas made this paper more fun to write. Dr. Diane Sorderholm, the Director of Education of the Bernard M. Gordon - MIT Engineering Leadership Program, helped us correct the key technical concepts in this paper. We also want to extend our thanks for Wolfram Research, who provided their Mathematica solution suite to all students and teachers in Knowledge City.

REFERENCES

[1] Lessig, L., Code and Other Laws of Cyberspace, Version 2.0 , Basic Books, 2006

[2] Koo H.Y., "A Meta-Language for Systems Architecting", Doctoral Thesis at Massachusetts Institute of Technology, 2005,

[3] Latour B., Science in Action: How to Follow Scientists and Engineers through Society, Harvard University Press, 1988.

[4] Baldwin C.Y. and Clark K.B., Design Rules, Vol 1: The Power of Modularity, MIT Press, 2000.

[5] Free Software Foundation, "About the Free Software Foundation", http://www.fsf.org/about, web page, last accessed on May, 24, 2009.

Biographical Information

Benjamn H. Y. Koo is an Associate Professor of the Department of Industrial Engineering at the Tsinghua University, Beijing, He is also a Special Consultant for the UNESCO's Chaired Professor on Industry and Academic Collaboration. His research interests are in the field of system design and management. His is currently working on a programming environment to represent and simulate the interactions between networked humans and machines. He served as the Chair for the first International Symposium on System Simulation and Education Technologies in China.

Professor Jianzhong Cha is the UNESCO Chair on Cooperation between Higher Engineering Education and Industries at Beijing Jiaotong University. He is also a Professor in Mechanical Engineering, Vice Chairman of the Academic Committee, and Vice Chairman of the Liaison Committee of University-Industry Cooperation at Beijing Jiaotong University. Professor Cha sits on the editorial board of eight academic journals, and has chaired more than 15 international conferences. He is Vice President of the Board of Directors of the Chinese Association of Intelligent Manufacturing, and a member of the board of directors of several academic organisations in and outside China.

Edward F. Crawley is the Ford Professor of Engineering in the Department of Aeronautics and Astronautics and a Professor of Engineering Systems at MIT. He is a Fellow of the Royal Aeronautical Society, the Royal Swedish Academy of Engineering Science, and the American Institute of Aeronautics and Astronautics (AIAA), and a member of the National Academy of Engineering. He has been awarded the NASA Public Service Medal.

Corresponding author

Dr. Benjamin H. Y. Koo
Tsinghua University
Shun De Building South #614
Beijing, Beijing, China 100084

+86 10 62792539
benkoo@tsinghua.edu.cn