Example of a Well-Designed Course

1. Specific Context

- The subject matter: *Computer Information Technology – Developer*
- The title of the course: *.Net Application Programming*
- Typical class size: *12-20*
- Level of the course: *First year, second semester*
- Mode of delivery: *Usually face-to-face; occasionally hybrid with 50% online*
- Type of institution: *Two-year community college having formal partnerships with state universities; mix of occupational and liberal arts programs*

2. General Description of the Course

Following a prerequisite course in programming logic and design, students learn and use computer language C# as they develop applications that run on Windows PCs. They plan and design new applications, implement and document their solutions in Visual Studio, and test their results on PCs. Students also begin applying object-oriented programming concepts which they will pursue much further in the following course.

Students create Windows applications

3. Big Purpose of the Course

Students are typically launching their chosen occupational study when they take this course; they seek a career directly aligned with it. Although the programming language itself may – and likely will – change, the thought processes and problem-solving approach they experience during class are very relevant for a long time.

This course also enables many students to determine whether they really are (or really are not) interested in a career in computer programming. It is very helpful to get clear on this important career issue early in their college education.

4. Important Situational Factors/Special Pedagogical Challenge

Students exhibit a broad variety of technical and social skills and motivation. Community employees seeking technical training are often very focused on content and have little tolerance of what they consider to be academic candy (i.e., learner-centered activities). Many of the traditional young-adult students received some programming training in high school; they typically coast through the first third of the semester and then have trouble shifting into learning / studying mode when new topics appear. Displaced adult workers seeking a new career and traditional students who did not study technology in high school are often shocked by the level of technical work; they request a slower pace to reduce stress – too slow for course success.

Occasionally an overly immature student attends – our institution offers open enrollment – and this adds stress to all students. Pace and behavior management issues appear and must be addressed promptly and directly.

Strong technical skills are critical: Visual Studio is one of the most complex computer applications students will ever see. In addition to using classroom workstations well, students must configure their own programming environment at home.

Many students bring stories of ineffective group work from high school, other college courses, or prior jobs; they object to being assessed in a major multi-part team project. However, our Advisory Committee directs us to implement more teamwork into our courses – they see collaborative weaknesses in new employees, and they (rightly) expect us to work on making major interpersonal breakthroughs as we teach.

Computer programming is a rapid-changing technology. New versions of the C# language appear every two to three years; hardware platforms, especially mobile devices, are replaced even faster. Thought-provoking, rich lessons are often replaced with built-in features and wizards in new software versions; many programmers publish programming solutions on the Internet too. Whereas learning outcomes can be fairly stable, textbooks and lessons are not.

5. 3-Column Table: Goals, Assessment, and Activities

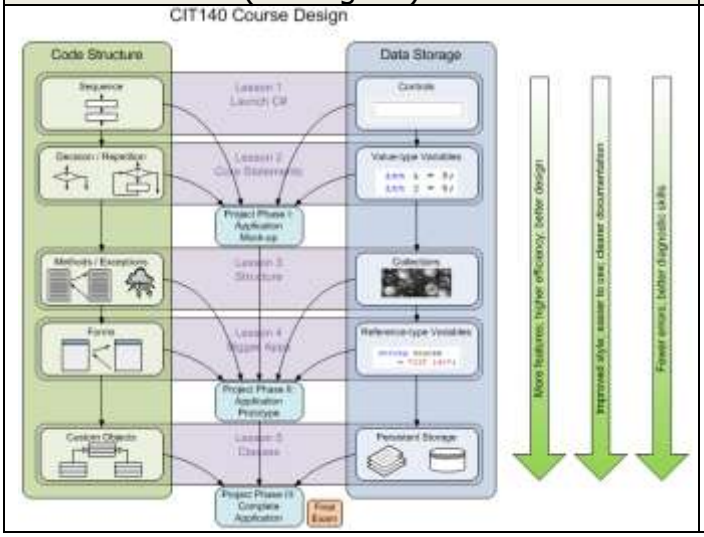
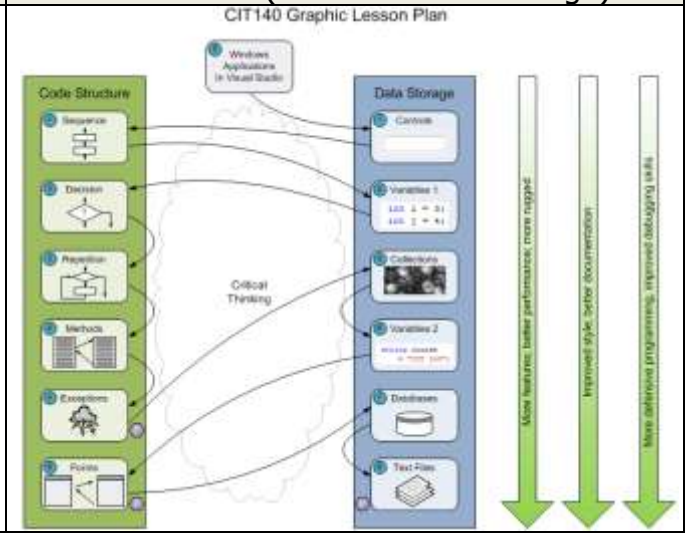
Learning Goals	Assessment Activities	Learning Activities
Recognize and understand common C# syntax and semantics (foundational knowledge)	*Online study guides (quizzes that students can repeat until right)*	*Read textbook; classroom guided practice*
Design, develop, test, and document custom C# Windows computer applications (application)	*Forward-looking individual C# homework assignments; team project; final exam*	*Forward-looking application design discussions; demos; code-writing activities; student demos*
Apply computer programming solutions to business and personal interests (integration)	*Forward-looking individual C# homework assignments; team project*	*Forward-looking application design discussions; demos; code-writing activities*
• *Discover personal interest in a career as an application developer (human dimension/Self)* • *Develop ability to perform effectively as a member of a work-team (human dimension/Other)*	*Reflective self-evaluation* *Peer feedback & evaluation*	*Discussions (classroom and/or forums); research* *Work on project teams*
Find passion to use computer programming technology to help people and society (caring)	*Team project; reflective self-evaluation*	*Classroom discussions; forums*
Learn how to learn about new [program codes] when they are [issued]. (learning how to learn)	*Treasure hunt homework assignments (students document their learning process)* *(Revise?)*	*Explanations; role-playing; practice treasure hunts; video demos* *(Revise?)*

The original syllabus (before course redesign) contained this explanation of learning outcomes:

> *CIT 140 is a computer programming course. Students are introduced to .NET application development using Visual Studio and the C# language. In CIT 140 you will:*
>
> - ***Design**, **develop**, **test**, and **document** Windows applications containing basic input and output, variables, arrays, menus, and integration with external data sources.*
> - ***Explore** and **practice** many aspects of object-oriented design and object-oriented programming.*
> - ***Discover** and **practice** extending your understanding of the .NET platform and the C# language through built-in help and on-line resources.*

The original approach focused primarily on content, a bit on approach. I had integrated discussions on personal connections during classroom sessions, but no formal learning goal (and no assessment) acknowledged its importance.

Graphic lesson plan

New (redesigned) version	Prior version (before course redesign)
	

Summary of changes:

- *Eight lessons were reorganized into five. This structural change improved lesson size and balance. One major technical concept was deemphasized; another was added.*
- *Strong relationships between lessons are now much clearer – no longer a simple sequential pattern.*
- *The three-phase project is shown now as a major course feature rather than as an occasional assessment technique (see three small hexagons with the letter "P" inside in the original version).*
- *The final exam is clearly acknowledged; a midterm exam was dropped.*
- *The explicit critical thinking "cloud" was removed from the diagram. That outcome now appears instead as a measured trait in grading rubrics, with explicit standards adding needed emphasis.*

6. Weekly Schedule of Learning Activities

Week	Before lesson	Mondays	Between classes	Wednesdays	After class, by Thursday a.m.
1	*Enroll; buy textbook*	*Course intro; write first C# program (Hello, World)*	*Configure Visual Studio on personal home PC or on laptop for future homework*	*Observations activity re: Windows programs; pre-class quiz on terminology*	
2	*Read; online L1 study guide*	*Design and implement Student profile application*		*Demo ListBox; Create calculator application*	
3		*Choose project topic; demo anchoring*	*(Launch homework)*	*Anchoring metaphor and application*	*Forward-looking programming assignment*

4	*Read; online L2 study guide*	*Expressions activity (formulas)*	*(Launch homework)*	*Catering planner (math expressions)*	*Self-reflection: programming for society / business*
5		*Flowchart demo; Tuition and fees flowchart and application*	*(Launch homework)*	*Loop Console application; ad hoc peer demo*	*Forward-looking programming assignment*
6		*Jing demo; project lab*	*Complete and upload project packet; practice presentation*	*Phase 1 team project presentations: app. mockup*	
7	*Read; online L3 study guide*	*Refactor L2 HW with methods; CIT grade method app.*	*(Launch homework)*	*Rugged data entry application*	*Treasure hunt: Programming references*
8		*Stacks and Queues*	*(Launch homework)*	*Collection looping application*	*Forward-looking programming assignment*
9	*Read; online L4 study guide*	*String manipulation: parse state abbrev.*	*(Launch homework)*	*Parse team name; Date/Time app: choice & peer demo*	*Self-reflection: career interest as programmer*
10		*Dialogs application*	*(Launch homework)*	*Interacting Forms application*	*Forward-looking programming assignment*
11		*Project lab*	*Complete and upload project packet; practice presentation*	*Phase 2 team project presentations: app. prototype*	
12	*Read; online L5 study guide*	*Class discussion; Digital Camera class*	*(Launch homework)*	*Custom class and application*	*Treasure hunt: Internet code solutions*
13		*Persistence discussion; Files and folders app*	*(Launch homework)*	*Text file storage application*	*Forward-looking programming assignment*
14		*Project lab*	*Complete and upload project packet; practice presentation*	*Phase 3 team project presentations: app. solution*	
15	*Proctored final exam outside class*				

Summary of changes:

- *I restructured checklist-style rubrics to grid-style rubrics with criteria and standards.*
- *Two reflective self-evaluation writing assignments (first time!) assess "soft" aspects of this technical course.*
- *Two Jing treasure hunt assessments (first time!) give evidence of students performing research.*
- *Automated study guides, repeated until students achieve desired grades; some immediate feedback is included; more feedback is available after deadlines arrive.*

- *There are now fewer demonstrations, more hands-on application activities in classroom (a minor change).*
- *There are fewer technical assignments. All were and remain forward-looking.*

Special considerations for this course:

- *All course materials other than the textbook are posted online on Moodle, our course management system, and students upload homework to and check grades on Moodle.*
- *Compiled programming assignments contain executable computer applications, a standard and desired outcome. However, most email filters block these files due to the risk of viruses, so I must provide an alternate way for students to request help asynchronously. I've chosen a Moodle forum dedicated to student help requests.*
- *In order to encourage participation, roughly 7% of the course grade is assigned to unchecked classroom activities. I allow students to make up missed participation points until the end of the semester.*

My philosophy of teaching and learning:

- *A teacher must be authentic, honest, and encouraging. He or she must set high expectations. He or she should be aware of personal bias and should strive to counteract it.*
- *A teacher should include learning in all activities - including assessment. (I do realize that some disagree with me.)*
- *A teacher needs to ensure that students really think - making sure they connect and are aware of content and of their learning.*
- *The environment must be safe: non-threatening, respectful, welcoming, and characterized by many low-risk practice activities.*
- *Learners need frequent assessment with timely feedback. They appreciate choice whenever it is feasible. Activities and content must engage learners; relevant project-based assignments help.*

7. Evidence of Impact

This redesigned course was first delivered fall 2010. Several aspects bubbled up:

- *Situational factors need adaptation: I did not expect adult professional students to be so resistant to collaborative and non-coding activities and assessments such as Jing video/screen capture and team projects.*
- *I was caught pleasantly off-guard by nearly universal acceptance of the two writing assignments. Also, after having taught the pre-redesigned version of the course twenty times, the insight gleaned from students' written self-reflections was truly enlightening! I added a similar assignment to a different course with similar, positive results.*

- *I am convinced that students in the redesigned course better understand the meaning and purpose of computer programming. The positive impact will be verified when students take the subsequent course.*
- *Based on final exam results and on office-hour visits, I do not believe students' programming skills were quite up to par. Planned improvements will leverage a technique that arose during critical analysis of the redesign. I expect technical skills to recover immediately.*

8. Most Exciting Aspect of the Re-Designed Course for You

I saw better engagement during class, more students helping each other during lab sessions. Attrition was lower than normal – more students stayed engaged until they successfully completed the course. Self-reflection writing assignments gave evidence of metacognition; I was never sure how effectively I was promoting critical thinking.

9. Your Contact Information

Your name: *Jeff Straw*

Your email address: *jstraw@nmc.edu*