PLATAFORMAS DE COMUNICACIÓN ACADÉMICA Y APRENDIZAJE COMBINADO

J. J. Carreón Granados[1]

RESUMEN

Cómo aprovechar plataformas de comunicación educativa como Piazza o MOOC formales e informales, mediante YouTube, buscando no sólo el fortalecimiento de asignaturas específicas, sino propiciando también entre los estudiantes el aprendizaje de temas de las asignaturas en idioma inglés. Al contar con valiosos videos, notas de clase, libros, software, material didáctico, tareas, exámenes, evaluaciones, proyectos y grupos de noticias, en este idioma, incluyendo cursos completos en Coursera. Los MOOC, conjuntamente con plataformas de comunicación, permiten a estudiantes preguntar, responder y consultar en promedio decenas y hasta cientos de contribuciones en relación con asignaturas y enriquecerlas con recursos en línea adicionales. No obstante los aparentes magros resultados iniciales de MOOC y plataformas como las mencionadas, en ciertas circunstancias estos resultados pueden ser ya significativos, pudiendo generar sinergias nada despreciables entre diferentes plataformas, asignaturas, campos, departamentos, dependencias, instituciones, regiones y culturas, por lo que se exponen algunas experiencias en esta dirección.

ANTECEDENTES

¿Cómo mejorar el aprendizaje y la enseñanza? ¿Cómo hacer más relevante lo que se aprende y enseña? ¿Cómo manejar recursos en inglés? ¿Cómo vincular el aprendizaje a la investigación? ¿Qué tan importante es elevar el nivel docente en asignaturas de Ingeniería de Software, en particular, Lenguajes de Programación?

Con base en la hipótesis de que Lenguajes de Programación es central a la Ingeniería de Software, ésta a la Ingeniería en Computación, la que a su vez es relevante para las Ingenierías, se ha planteado como objetivo elevar el nivel de la asignatura de ese nombre, clave 1671, con tres horas teóricas por semana, impartida en el sexto semestre a estudiantes de Ingeniería en Computación en la División de Ingeniería Eléctrica, FI, UNAM. (Facultad de Ingeniería, UNAM, 2010)

A fin de alcanzar este objetivo, se busca aprovechar recursos más avanzados, actualizados, accesibles y completos de colegas del Programming Languages Team, PLT, (Racket, 2014) con los que se ha trabajado a lo largo de más de una década. En particular, los cursos CS 5510: Programming Languages, Fall 2013, impartido bajo la modalidad de *salón invertido* por Matthew Flatt, School of Computing, University of Utah, EUA, (University of Utah, 2013); el CSCI 1730: Introduction to Programming Languages, en línea por Shriram Krishnamurthi, Brown University, EUA, (Brown University, 2012); y, en el MOOC (MOOC.ES, 2013) Coursera (Coursera, 2013) dos ediciones de Introduction to Systematic Program Design – Part 1, uno iniciado en junio y otro en septiembre de 2013, por Gregor Kiczales, The University of British Columbia, Canadá.

Como un antecedente importante, es necesario mencionar que ya en 2012 se aprovecharon recursos del CSCI 1730: Introduction to Programming Languages (Brown University,

[1] Profesor Titular de Tiempo Completo. Facultad de Ingeniería de la Universidad Nacional Autónoma de México. juan.carreon@gmail.com

2012), el cual se ofreció como experimento de curso combinado y puramente en línea empleando Piazza, (Piazza, 2012), entre otros recursos.

Como parte de los recursos empleados se dispuso del texto Cómo Diseñar Programas (Felleisen, Findler, Flatt, & Krishnamurthi, Cómo diseñar programas, 2009); traducción al español de How to Design Programs, HtDM, (M. Felleisen, 2003), accesible gratuitamente en línea; igual que el borrador de la segunda edición (Felleisen, Findler, Flatt, & Krishnamurthi, 2014); asimismo, del texto Programming Languages: Application and Interpretation, PLAI, versiones 2007 y 2012, Brown University, 2012 (Krishnamurthi, Programming Languages: Application and Intepretation, Second Edition, 2012); todos los cursos mencionados acompañados del software correspondiente, libremente accesible (PLT P. L., 2014).

Se ha aprovechado también la circunstancia de que los grupos en que se impartió la asignatura Lenguajes de Programación, LdP, fueron asignados a estudiantes que desearan cursar asignaturas con mayores contenidos en idioma inglés.

Los objetivos de ese curso teórico, son:

> "el alumno explicará la importancia de estudiar las características y paradigmas de los lenguajes; además podrá discernir, de entre los diferentes lenguajes existentes, los óptimos para desarrollar sistemas de software de alta calidad; diseñara nuevos lenguajes para computadora".

En un temario que abarca:

> 1) Fundamentos generales de los lenguajes de programación, 6 horas. 2) Principios de diseño de lenguajes, 6 horas. 3) Tipos de datos, 6 horas. 4) Expresiones y declaraciones, 4.5 horas. 5) Procedimientos y entornos, 4.5 horas. 6) Paradigmas de lenguajes, 21 horas.

Por su parte, el experimental CSCI 1730 (Brown University, 2012) mencionado previamente se orientó no tanto a la sintaxis de los lenguajes de programación, sino a su semántica, mediante construcción y validación de lenguajes; no sólo matemáticamente (como se acostumbra en la investigación en lenguajes de programación), sino desde una perspectiva ingenieril. Estando disponibles casi todos sus recursos: libros, videos, material complementario, software, grupos de noticias en Piazza y Google, así como evaluaciones, 26, para la parte teórica; dos proyectos y diez tareas para la parte práctica. Además de cinco tareas generales acerca de conceptos vistos en el curso.

Si bien los objetivos y contenidos de ambos cursos difieren, los de Lenguajes de Programación se cubrieron más que satisfactoriamente; los de Introduction to Programming Languages sólo se cubrieron un poco menos de la mitad de sus objetivos teóricos, los que en todo caso fue bastante eficaz, baste decir que el tema 6) Paradigmas de lenguajes, 21 horas, se cubrió mejor en la primera sesión de una hora en el CSCI 1730. (Krishnamurthi & Gibbs Politz, Programming Languages - Lecture 1, 2012)

En la segunda mitad de 2013, a fin de superar la falla de ver sólo aspectos teóricos y de forma incompleta, e incidir en deficiencias en la capacidad de programación de los

estudiantes, y teniendo entonces disponible el curso en línea Introduction to Systematic Program Design – Part 1, ISPD, de Gregor Kiczales, (Coursera, 2013) cuyo curso de nueve semanas coincidió con el semestre lectivo 2014-1 de Lenguajes de Programación, se decidió cursarlo en paralelo con éste último.

Los diez módulos del curso de Kiczales fueron:

> 1) Beginning Student Language, How to Design Functions; 2) How to Design Data; 3) How to Design Worlds; 4) Self-Reference, Reference; 5) Naturals, Helpers; 6) Binary Search Trees, Mutual Reference; 7) 2-One-Of, Local; 8) Abstraction; 9) Generative Recursion, Search; 10) Final exam.

El foco eje de este curso fue comprender y aplicar el proceso de diseño sistemático de programas. Debido a que los lenguajes, sean intérpretes o compiladores, son programas, centrarse en la semántica de éstos más que en su sintaxis, permite no sólo comprender los principios de los lenguajes de programación, sino incrementar la capacidad de manejarlos adecuadamente. A pesar de que parecería que este curso de Coursera tendría un contenido teórico menor al de Lenguajes; la experiencia de participación en él fue desafiante, enriquecedora y única, para muchos de sus participantes, incluyendo miembros del propio equipo de investigación PLT.

Pues, si bien se pensó para estudiantes que programaran poco o nada, los que en el LdP local y en el ISPD global representaron la gran mayoría de los participantes, sin embargo, desde el inicio en los foros del ISPD se trataron temas avanzados de lenguajes de programación como: efectos-laterales, *lambdas*, *closures*, y comparaciones sofisticadas de lenguajes, entre otras. Temas que pudieron compartimentarse en estos foros eficaz y eficientemente mediante la etiqueta *#gettingahead*. (Coursera, 2013)

Cada módulo consistió en cerca de una docena de videos, mejor y más elaborados que los del experimental CSCI 1730, los que también podían consultarse en línea o descargarse, con opciones de manejo de subtítulos de buena calidad, muchos incluyendo pequeñas pruebas incorporadas; cada módulo con más de docena y media de problemas de práctica, con sus respectivas soluciones; así como, al final una prueba de comprensión.

El curso incluyó, también, tres proyectos evaluados entre pares. Así como la opción *Signature Track* (Coursera, 2013) que permitió vincular la calificación en el curso a la identidad personal, y recibir un Certificado Verificado de la universidad de British Columbia. La que promete ofrecer próximamente nuevamente este curso, ISPD, así como su continuación. En los círculos internacionales más destacados de la docencia y la investigación de Lenguajes de Programación, existiría un cierto consenso de que ISPD elevó las cotas de eficacia y eficiencia, no sólo de lo que puede ser un buen MOOC, sino un curso excelente del área; al que para bien o para mal se parece demasiado.

Un poco por esta circunstancia, y mucho debido a nuevas perspectivas abiertas por el Blended Learning, Aprendizaje Mezclado, (Wikipedia, 2014) en su opción de Flipped Class, Clase Invertida, (Knewton, 2013) Matthew Flatt organizó como clase invertida el curso CS 5510: Programming Languages, Fall 2013, School of Computing, University of

Utah, EUA. Apoyando sus clases, en recursos más modestos, pero muy eficaces, al grabarlas como *screencasts* (como grabaciones digitales de la salida por pantalla de la computadora, acompañadas por narración de audio), *pdfs* de presentaciones, ejercicios y tareas. (Flatt, 2013)

Flatt más que buscar llegar a grandes grupos de estudiantes, experimentó nuevas formas de organizar su clase, aproximadamente 40 estudiantes, mediante grabaciones de video y recursos en línea, organizados en línea a lo largo de una década de docencia e investigación en el área de Lenguajes de Programación. Como libro de texto empleó también, Programming Languages: Application and Interpretation de Shriram Krishnamurthi, en su segunda edición. (Krishnamurthi, Programming Languages: Application and Intepretation, Second Edition, 2012).

CS 5510 Programming Languages es un curso tradicional en Lenguajes de Programación, con base en el diseño de intérpretes. En la que los estudiantes emplean *plai-typed* Racket (PLT P. L., 2014) a fin de implementar intérpretes y verificadores de tipos, con la idea de que instrumentar constructos de lenguajes de programación conduce a una comprensión sólida de constructos en lenguajes actuales y futuros. De ahí que un aspecto importante del curso fuera que los estudiantes realizaran una tarea semanal en la forma de "a este intérprete agregarle el rasgo X", esperando que los estudiantes invirtieran del orden de diez horas a la semana en este desarrollo.

Para cada tarea, dos o tres estudiantes presentaban sus soluciones al resto de la clase. Cada uno teniendo que presentar cuando menos una vez su solución durante el semestre, estas soluciones no tendrían que ser completas, pues se hacían antes de la fecha de entrega, en lapsos de entre 20 y 30 minutos. El resto dedicándose a una presentación de los temas por el profesor con base en transparencias elaboradas y mejoradas minuciosamente a lo largo de los años. Ahora las transparencias han sido editadas bajo la forma de videos de entre tres y ocho minutos, los que se supone han sido vistos antes de clase, dedicándoles ahora entre 20 y 40 minutos a la presentación de tareas en clase, pues se ha reducido la presentación de material en la misma.

En cambio, se dedica más tiempo a trabajar con ejemplos adicionales, lo que significa modificar un intérprete y/o un verificador de tipo. A veces, el profesor manejando el teclado, de preferencia es un estudiante el que lo maneja, no se espera que los estudiantes resuelvan ellos solos los ejercicios, pero sí que pidan ayuda al resto de la clase. Muchas veces, después de entre 5 y 10 minutos, un estudiante diferente se sienta frente al teclado. Clases mediante videos coincide con la definición de clase invertida, *flipped classroom*, (Knewton, 2013) en la que se invierten los métodos tradicionales de enseñanza, impartiendo clases en línea fuera de clase, y resolviendo la "tarea" en el salón de clase.

METODOLOGÍA

En el contexto descrito, ¿podría ser el aprendizaje mezclado una mejor opción, bajo la forma de clase invertida?

Las clases como videos coinciden con lo que hace Flatt; un poco menos, con Kiczales, y, aun un poco menos con Krishnamurthi, aunque explícitamente lo menciona como tal. Flatt

cree que “el salón de clase invertido”, si significa que los estudiantes terminen su tarea en el salón de clase, en donde el profesor ayuda individualmente a los estudiantes según lo requieran. Sin embargo, para él requiere que los estudiantes inviertan 10 horas a la semana en avanzar en su tarea, y puesto que se reúne con ellos menos de tres horas a la semana, en sentido riguroso no se apega a la definición de esa forma de enseñanza-aprendizaje.

Su curso también sigue siendo tradicional en el sentido de que en la medida en que la clase avanza a través del material, en una perspectiva que es “ver y revisar los videos cada quien a su propio ritmo”, pero no trabajar cada quien a su propio ritmo. Sin embargo, es un salón de clase invertido en el sentido que trabajando en un ambiente universitario, en donde la clase se centra en lo que alguien, principalmente un estudiante, dice o hace. Sin embargo, el aspecto central es que el supuesto es que los estudiantes consulten el material fuera de clase, y no invertir tiempo de clase en revisarlo.

La principal tarea de ese curso ahora fue grabar videos, los cuales seguramente serán y son reusados; esa tarea es un orden de magnitud superior a cualquier otra demandada por el curso, pero un orden de magnitud inferior a si el curso fuera MOOC. Flatt considera que el CS 5510 revisado es el curso más satisfactorio de su carrera, cubriendo más material y a mayor profundidad, los alumnos en un 30% cumplieron más con las tareas, igual sucedió con las calificaciones y los estudiantes aprobados.

Con base en esos resultados, él recomienda manejar videos pequeños, de menos de 10 minutos; resistir la tentación de retomar y resumir los videos en clase; hacer que los estudiantes muestren sus tareas en clase, incluso antes de la fecha establecida para entregarla. Elaborar ejemplos adicionales en el último minuto antes de la clase, resultó con efectos contrarios a lo que tradicionalmente sucedía; darle el gis, el teclado, o lo que sea a un estudiante, aun si la clase entera tiene que decirle al estudiante cada letra, es mucho mejor que uno como profesor hacerlo.

Pues, la inversión según la infografía “The Flipped Classroom” invierte la relación de “sabio en el podio” a “guía al lado”; el modelo implica que los estudiantes atiendan las clases en sus casas a su propio ritmo, comunicándose con pares y profesores mediante discusiones en línea; el aprendizaje conceptual se produce en el aula con ayuda del instructor; el aula invertida propicia oportunidades para aprender mediante actividades, empleando tecnología educativa, influyendo ambas en el ambiente de aprendizaje. Lo que en algunos casos ha generado resultados más que significativos: de índices de reprobación de 50% en inglés y 44% en matemáticas, se ha pasado a índices de 19% y 13%, respectivamente. (Knewton, 2013)

En el caso de Lenguajes de Programación de una forma no tan clara, pero si con objetivos como los mencionados previamente: ¿Cómo mejorar el aprendizaje y la enseñanza? ¿Cómo hacer más relevante lo que se aprende y enseña? ¿Cómo manejar recursos en inglés? ¿Cómo vincular el aprendizaje a la investigación? ¿Qué tan importante es elevar el nivel docente en asignaturas de Ingeniería de Software? ¿En particular, en Lenguajes de Programación? Se comenzaron a emplear videos, entre otros materiales de autoaprendizaje, pero sin tener claro que podrían implicar una forma diferente de aprendizaje que en sí fuera más efectiva que la tradicional. No fue sino hasta la primera parte del 2014 que esto

empezó a ser evidente. De ahí los diferentes resultados obtenidos en los últimos dos años, así como la poca conciencia de las limitaciones y alcances implícitos en la nueva forma de trabajar, cuyos alcances son mucho más amplios de lo que parecían inicialmente.

DISCUSIÓN DE RESULTADOS

El primer y segundo semestres impartidos con la nueva perspectiva el foco de atención fue la excelencia de los materiales, en cambio, no se atendió suficientemente el darle la importancia y el trato adecuado al trabajo que de forma específica tendría que haberse dado en clase. Sin embargo, algunas herramientas y materiales indispensables comenzaron a emplearse, p e, Piazza, (Piazza Technologies, 2014) YouTube, Coursera, presentaciones, ejercicios, tareas, y proyectos.

Fueron notables los avances que se dieron empleando plataformas de comunicación académica como Piazza y BitácoraFI, (FI, UNAM, 2013) ésta una herramienta orientada a la comunicación con estudiantes del primer semestre en la FI, UNAM.

Estas herramientas permitieron a los estudiantes preguntar, responder y consultar en promedio decenas, si no es que cientos, de preguntas en relación a asignaturas y enriquecerlas significativamente con valiosos recursos en línea adicionales.

Al respecto, Piazza ha facilitado medir a grandes rasgos la participación en línea de un grupo, qué tanto los estudiantes expresan sus dudas, el tiempo promedio de respuesta de los estudiantes a preguntas de los propios estudiantes, quiénes contribuyen más, quiénes dan mejores respuestas (Share Your Class, Piazza, 2014).

Al respecto, en la Figura 1, se muestra el comportamiento de los estudiantes del curso CSCI 1730: Introduction to Programming Languages (Brown University, 2012), casi típico de muchos MOOC, donde después de un entusiasmo desbordante inicial, la participación cae aceleradamente; en la Figura 2, el de Cultura y Comunicación 1107, 1er semestre, Ciencias Sociales y Humanidades, asignatura socio humanística, Ingeniería Geofísica (FI, UNAM, 2006), se muestra la evolución típica de estudiantes de primer semestre en la FI. En la Figura 3, el de Recursos y Necesidades de México, materia común a todas las carreras de Ingeniería en la FI, (División de Ciencias Sociales y Humanidades, 2006) en donde en su segunda parte se aplicó “Flipped Classroom” (Knewton, 2013).

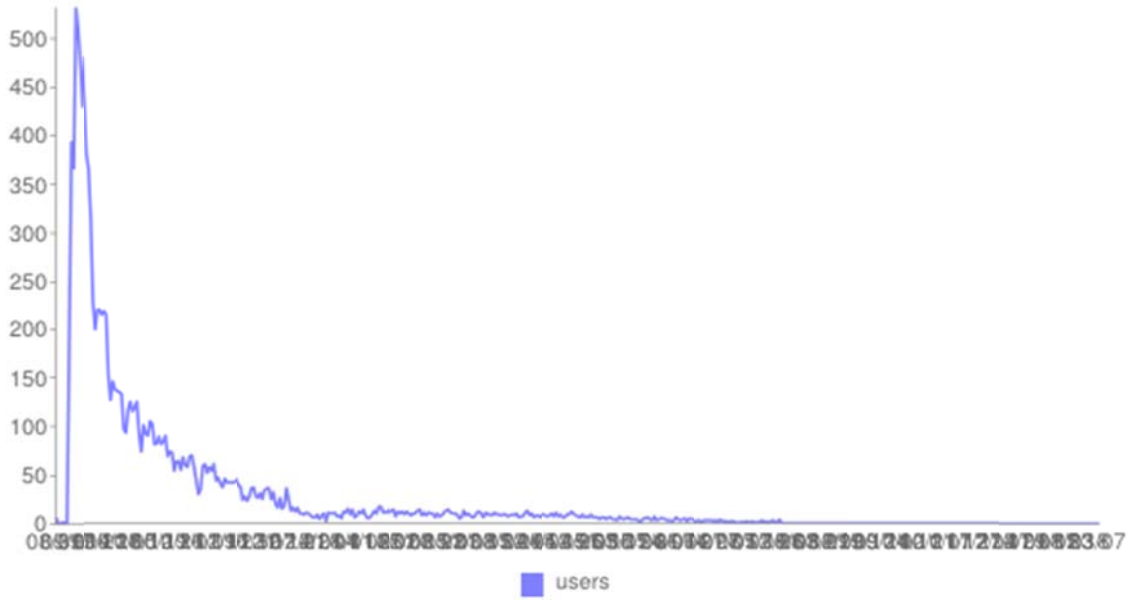

Figura 1. Comportamiento de CSCI 1730: Introduction to Programming Languages

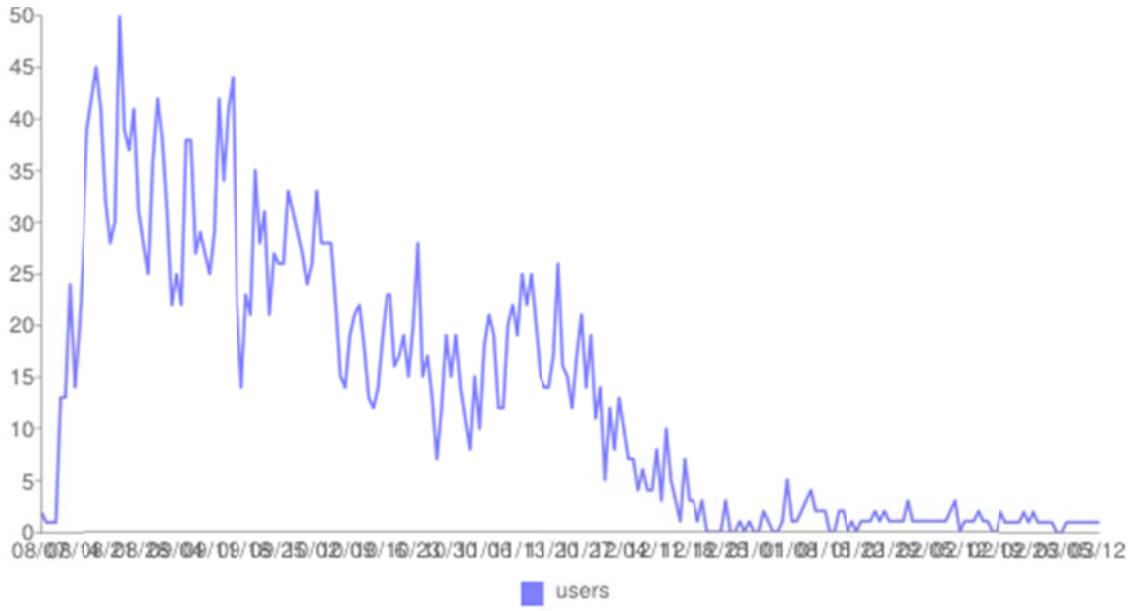

Figura 2. Comportamiento Cultura y Comunicación 1107, 1er semestre

No es de sorprender que algunos de los resultados más gratificantes fueran en asignaturas alejadas de Lenguajes de Programación; p e, algunas ideas como emplear herramientas de comunicación académica en línea como BitácoraFI, Piazza, YouTube, Coursera, KhanAcademy, e incluso Facebook, funcionaron bien para apoyar el aprendizaje de materias socio-humanísticas de estudiantes de primer semestre de Ingeniería Geofísica, (FI, UNAM, 2006) así como para contribuir a que superaran deficiencias en el área de Matemáticas, abriendo una promisoria línea de investigación orientada a elevar la calidad

de la docencia de las ingenierías. Con más conciencia de lo que se está haciendo y cómo se está haciendo, resultados parecidos a los mencionados también pueden observarse en otras asignaturas. (División de Ciencias Sociales y Humanidades, 2006)

CONCLUSIONES

Disponer de recursos didácticos adicionales de gran calidad demanda atender cuidadosamente no sólo ciertos aspectos de esos recursos, sino priorizar aspectos de la educación presencial que muchas veces no son suficientemente atendidos de forma adecuada, como tutorías y asesorías personalizadas, además de otras actividades presenciales adicionales. Las cuales inicialmente pueden parecer desproporcionadas con respecto a los resultados. Lo que podría estar sucediendo es que quizá no obstante el avance que representa el aprendizaje combinado choca con límites marcados al emplear nuevos medios en ambientes académicos demasiado convencionales, los que si también se modifican los resultados mejoran significativamente, como se muestra en la parte final de la Figura 3.

No obstante los aparentes magros resultados iniciales de MOOCs y plataformas como las mencionadas, estos resultados en ciertas circunstancias pueden ser ya significativos. Pudiendo generar sinergias nada despreciables entre diferentes plataformas, asignaturas, campos, departamentos, dependencias, instituciones, regiones y culturas.

Figura 3.Empleando "salón invertido" en Recursos y Necesidades de México

BIBLIOGRAFÍA

Brown University. (2012). *CSI 1730 Introduction to Programming Languages*. Obtenido de http://cs.brown.edu/courses/cs173/2012/

Brown University. (2012). *Krishnamurthi, S. CSCI 1730: Introduction to Programming Languages, Fall 2012*. Obtenido de

http://cs.brown.edu/courses/cs173/2012/

Coursera. (2013). *Earn a Verified Certificate*. Obtenido de https://www.coursera.org/signature/course/religionandtolerance/971337?utm_source=catalog

Coursera. (September de 2013). *Introduction to Systematic Program Design - Part 1 by Gregor Kiczales*. Obtenido de https://class.coursera.org/programdesign-002/forum/list?forum_id=8

Coursera. (September de 2013). *Kiczales, G, Introduction to Systematic Program Design – Part 1, September 2013*. Obtenido de The University of British Columbia: https://www.coursera.org/course/programdesign

División de Ciencias Sociales y Humanidades. (2006). *Recursos y Necesidades de México*. Obtenido de http://www.dcsyhfi.unam.mx/index.php?option=com_content&task=view&id=129&Itemid=155

Facultad de Ingeniería, UNAM. (2010). *Plan de Estudios de la carrera de Ingeniería en Computación*. Obtenido de http://www.ingenieria.unam.mx/paginas/Carreras/planes2010/Computacion/06/lenguajes_de_programacion.pdf

Felleisen, M., Findler, R. B., Flatt, M., & Krishnamurthi, S. (27th de March de 2014). *How to Design Programs, Second Edition*, 2nd Edition. (M. Press, Editor) Recuperado el 7 de abril de 2014, de http://www.ccs.neu.edu/home/matthias/HtDP2e/

FI, UNAM. (2006). *Plan de Estudios de la Carrera de Ingeniería Geofísica*. Obtenido de http://www.ingenieria.unam.mx/paginas/Carreras/planes2010/Geofisica/01/cultura_y_comnunicacion.pdf

FI, UNAM. (2013). *Aprendizaje autónomo*. Obtenido de http://www.ingenieria.unam.mx/~bitacoraFI/bitacora/index.php

Flatt, M. (5 de December de 2013). *I Flipped a Class, and I Liked It. Blog: Unclosed Parenthesis, Thursday, December 5, 2013*. Obtenido de Blog: Unclosed Parenthesis, Thursday, December 5, 2013: http://unclosedparenthesis.blogspot.mx/2013/12/i-flipped-class-and-i-liked-it.html

Knewton. (2013). *Flipped Classroom*. Obtenido de http://www.knewton.com/flipped-classroom/

Krishnamurthi, S. (2012). *Programming Languages: Application and Intepretation, Second Edition*. Obtenido de

http://cs.brown.edu/~sk/Publications/Books/ProgLangs/2007-04-26/

Krishnamurthi, S., & Gibbs Politz, J. (2012). *Programming Languages - Lecture 1*. Obtenido de https://www.youtube.com/watch?v=3N__tvmZrzc

M. Felleisen, M. e. (2003). *How to Design Programs, September 2003*. Obtenido de http://htdp.org/

MOOC.ES. (2013). *¿Qué es un MOOC?* Obtenido de http://www.mooc.es/que-es-un-mooc/

Piazza. (2012). *PL Online - Fall 2012*. Obtenido de PL 1: Programming Languages: https://piazza.com/pl_online/fall2012/pl1/home

Piazza Technologies. (21 de marzo de 2014). *Piazza*. Obtenido de https://piazza.com/signup

PLT, P. L. (2014). *Programming Language Download*. Obtenido de http://cs.brown.edu/courses/cs173/2012/lang/

PLT, P. L. (7 de abril de 2014). *Racket*. Obtenido de http://racket-lang.org/

Racket. (2014). *Community*. Obtenido de http://racket-lang.org/

Share Your Class, Piazza. (7 de abril de 2014). *Class at a Glance*. Obtenido de https://piazza.com/demo_login?nid=hju7zywmtmj7mn&auth=46ceb8c

University of Utah. (2013). *Flatt, M. CS 5510: Programming Languages, Fall 2013*. Obtenido de http://www.eng.utah.edu/~cs5510/

Wikipedia. (20 de marzo de 2014). *Blended learning*. Obtenido de http://en.wikipedia.org/wiki/Blended_learning