Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Análisis comparativo de propuestas MOOC sobre Ingeniería del Software

ESCUELA POLITECNICA
SUPERIOR

Autor: Alejandro Palomo Díaz-Flores

Tutor/es: Carmen Pagés Arévalo

Junio 2016

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Análisis comparativo de propuestas MOOC sobre Ingeniería del Software

Autor: Alejandro Palomo Díaz-Flores

Tutor/es: Carmen Pagés Arévalo

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

FECHA:

Índice

RESUMEN 4

SUMMARY 4

PALABRAS CLAVE 4

INTRODUCCIÓN 5

OBJETIVOS DEL PROYECTO 5

ESTADO DEL ARTE 6

Ingeniería del Software 6

MOOC 8

MOOC SOBRE INGENIERÍA DEL SOFTWARE 12

Coursera 12

Desarrollo y Diseño de Videojuegos 13

Gestión de Producto de Software 14

Desarrollo de aplicaciones móviles con Android 15

Drive to Value with Agile Methods 15

Udacity 16

Software Development Process 16

Software Testing 21

Intro to Hadoop and MapReduce 24

EdX 26

Developing International Software 26

Agile Development Using Ruby on Rails 30

Computing: Art, Magic, Sciencie 31

Preparing for CLEP Information Systems Exam 34

Enterprise Software Lifecycle Management 36

Introduction to Programming with Java Part 2: Writing Good Code 38

Miríada X 39
Agilidad y Lean. Gestionando los proyectos y negocios del s. XXI 40
MOOC de otras plataformas 43
Software Engineering 43
Pex4Fun 44
RESUMEN Y CONCLUSIONES 45
BIBLIOGRAFÍA 50

RESUMEN

A largo de este documento se realizará un análisis de los MOOC relacionados con la Ingeniería del Software albergados en las principales plataformas donde se ofrecen, del material docente, modos de evaluación y métodos de enseñanza que ofrecen cada uno.

El análisis del material docente se centrará en el temario relacionado con la Ingeniería del Software, es decir, todo lo relacionado con metodologías, técnicas... Dentro de este temario, se profundizará en el tipo de material y modo de docencia (vídeos, textos, ejercicios…), y en el modo de evaluación (test, exámenes…)

SUMMARY

This document provides an analysis about MOOCs related to Software Engineering held in the main platforms where they are offer is made, about their teaching material, ways of evaluation and teaching methods offered by each of them.

The analysis of the teaching material will be focused on the agenda related to the Software Engineering, i.e., everything related to methodologies, techniques… Within this agenda, it will deepen the material kind and the teaching mode (videos, texts, exercises…), and the way of evaluation (test, exams…)

PALABRAS CLAVE

MOOC, Ingeniería del Software.

INTRODUCCIÓN

Los MOOC (en inglés Massive Open Online Courses, es decir, Cursos Online Masivos Abiertos) son cursos online ofrecidos por distintas instituciones abiertos a un gran número de posibles participantes por medio de Internet.

Entre los puntos fuertes de los MOOC, se podrían destacar principalmente el número ilimitado de matriculaciones en un curso, la capacidad de poder atenderlo online y el ser gratuito y accesible, tanto el curso como el material didáctico necesario. Aunque hay muchos MOOC que dependiendo de la institución no son gratuitos y hay que pagar para poder seguirlo.

Con el auge global de Internet, estos tipos de cursos han experimentado un gran crecimiento desde su nacimiento en el año 2008. Con esta crecida, el número de MOOC ofrecidos y el número de plataformas que lo ofrecen es cada vez mayor. Cada vez más universidades y entidades docentes incluyen MOOCs entre sus servicios ofrecidos.

En este documentos se han diferenciado, dentro de esas plataformas, los MOOC relacionados con la Informática en general de los MOOC relacionados con la Ingeniería del Software en particular. Dentro de estos últimos, se pueden establecer tres grupos de MOOC: los MOOC relacionados con la programación exclusivamente, es decir, que se centran en la programación y no incluyen ningún otro ámbito de la Ingeniería del Software; los MOOC relacionados con la programación que incluyen otras fases del desarrollo de programación (diseño, metodologías, pruebas, diseño, etc.); y los MOOC centrados en técnicas, fases, metodologías… del proceso de desarrollo del software, que incluyen poco o nada de temario relacionado con la programación. Por tanto, se puede establecer una clasificación en tres grupos: los MOOC relacionados con la Informática, los MOOC relacionados con la programación que incluyen otras fases del desarrollo del software, y los MOOC orientados a técnicas o fases de la Ingeniería del Software.

OBJETIVOS DEL PROYECTO

Este desarrollo se va a centrar principalmente en los MOOC orientados a la Ingeniería del Software, estudiando distintos cursos que los ofertan y analizando la manera que se tiene de impartirlos.

A lo largo de este desarrollo se va a centrar el análisis en los MOOC relacionados con la Ingeniería del Software, estudiando tanto MOOC de programación/desarrollo en general que contengan en su temario alguna parte relacionada con la Ingeniería del Software como los MOOC enteramente dedicados a ella (Estos últimos serán los más interesantes desde el punto de vista de este estudio).

Este estudio consistirá tanto en un análisis cuantitativo como cualitativo, es decir, la cantidad de MOOC de cada plataforma que están dedicados a la Ingeniería del Software o contienen algo de temario sobre ella; qué cantidad de temario y la calidad de ese temario. Se procederá a analizar el temario relacionado con la Ingeniería del Software de cada MOOC para comprobar que temas trata y cómo los aborda.

Por tanto, el objetivo en cada curso es establecer qué partes del temario corresponden a la Ingeniería de Software, analizar con profundizad ese temario, establecer cómo se organiza y cómo se evalúa.

ESTADO DEL ARTE

Ingeniería del Software

La Ingeniería del Software puede definirse como el uso y el establecimiento de las técnicas relacionadas de la ingeniería en el ámbito de un desarrollo software con el objetivo de que sea confiable y eficiente. Es una tecnología de varias capas, que combina un estilo de organización con un aseguramiento de la calidad.

Siguiendo este objetivo, la Ingeniería del Software está siempre en constante mejora y evolución, en busca de enfoques más eficaces para lograrlo. Las capas que componen la Ingeniería del Software son el compromiso de calidad, el proceso, los métodos, y las herramientas.

Los métodos de la ingeniería del software son un conjunto de principios que dirigen las áreas y las actividades de la elaboración del software, es decir, define las tareas relacionadas con la comunicación, el análisis de los requisitos, el modelado, las pruebas…

Las herramientas de la ingeniería del software suponen un apoyo a los métodos y procesos con el fin de intentar automatizar e integrar alguna de sus tareas.

El proceso de la ingeniería del software representa el conjunto de actividades o tareas que se van realizando a la hora de crear el producto software. En este ámbito, el proceso es adaptable según el equipo de desarrollo considera para poder satisfacer a los clientes, y asegurar la calidad deseada. El proceso se

estructura por medio de actividades de tal manera que sean aplicables a cualquier desarrollo de software independientemente de su tamaño o complejidad. Esta estructuración consta de cinco actividades:

- Comunicación: esta actividad abarca la relación con el cliente, con el objetivo de entender y especificar mediante requisitos el producto deseado por el cliente.
- Planificación: consiste en la descripción de las tareas a realizar a lo largo del desarrollo, los recursos necesarios y los riesgos a tener en cuenta.
- Modelado: esta actividad consiste en el diseño de la aplicación, una visión modelada de lo que se requiere conseguir.
- Construcción: principalmente, se centra en la implementación del código en sí y en las pruebas necesarias para descubrir en errores en él.
- Despliegue: Una vez completado el software, se entrega al cliente para que lo evalúe.

Estas actividades pueden variar en contenido y detalles según la complejidad y el tamaño del desarrollo, pero la estructura siempre será la misma. Además de estas actividades principales, a menudo se complementan con otras actividades como:

- Seguimiento y control del proyecto de software: mediante esta actividad, el equipo de software puede evaluar el progreso del desarrollo.
- Administración del riesgo: consiste en evaluar los riesgos que puede afectar al resultado del proyecto o a la calidad final del producto.
- Aseguramiento de la calidad del software: define las actividades necesarias para garantizar la calidad del software.
- Revisiones técnicas: consiste en evaluaciones periódicas del producto con el objetivo de descubrir y evaluar errores y evitar que se propaguen a las siguientes actividades.
- Medición: mide los progresos del proceso, del proyecto y del producto con el fin de ayudar al equipo a valorar estos progresos.
- Administración de la configuración del software: abarca los posibles cambios a lo largo del proceso.
- Administración de la reutilización: consiste en volver a utilizar del producto realizado, o definir mecanismos para obtener componentes reutilizables.
- Preparación y producción del producto del trabajo: abarca todo lo relacionado con la creación del producto (modelos, documentos, registros, formatos y listas).

Al conjunto de todas las etapas anteriores se le denomina Ciclo de Vida del desarrollo del Software. Existen varios tipos de ciclos de vida en función de cómo establecen el proceso de desarrollo de un producto software:

- Modelo en cascada: es el modelo clásico habitual, con un enfoque secuencial que comienza con la especificación de requisitos y continúa avanzando por la planificación, el modelado, la construcción y el despliegue. Es el enfoque clásico, pero en los últimos tiempos ha ganado una gran cantidad de detractores en beneficio de otros modelos más modernos.
- Modelo evolutivo: este modelo está pensado para desarrollos de productos que evolucionan en el tiempo, es decir, es un modelo diseñado para adaptarse a cambios. Existen principalmente dos modelos evolutivos:

    - Mediante prototipos: se basa en la construcción de un prototipo con las funcionalidades básicas para presentarse al cliente, y a partir de ese prototipo se prosigue el desarrollo.
    - Modelo espiral: cada iteración representa un "bucle" de una espiral en el que, conforme se vaya avanzando en esa espiral, las actividades que conforman la iteración se van volviendo cada vez más complejas.
- Modelo iterativo e incremental: este modelo organiza el desarrollo en pequeños incrementos o iteraciones, cada una de las cuales es un proceso en sí, y al finalizarlas se obtienen resultados entregables al cliente.
- Modelo ágil: es un modelo pensado para adaptarse a los cambios y a las situaciones impredecibles, por tanto, este modelo se caracteriza por la capacidad de ser adaptable a los cambios que se vayan produciendo a lo largo del proyecto. En cada iteración se encapsula un ciclo de vida completo, y se hace especial hincapié en la comunicación. Entre los modelos ágiles más utilizados destacan la Programación Extrema (XP) y SCRUM.
    - Programación Extrema (eXtremme Programming/XP): en este modelo destacan los valores de la simplicidad, la comunicación, la retroalimentación, la valentía y el respeto. Este modelo es el más utilizado de los modelos ágiles. En este modelo destacan la realización de pruebas unitarias continuas, el desarrollo iterativo e incremental, la programación en parejas, la refactorización del código, la corrección de todos los errores antes de empezar con una nueva funcionalidad o la simplicidad en el código.
    - SCRUM: este modelo ha demostrado su eficacia en proyectos con plazos de entrega apretados, requisitos cambiantes y negocios críticos. Acentúa el patrones de proceso de desarrollo de software como los "sprint", que son unidades de trabajo en los que se define una cantidad de trabajo que debe ajustarse a la duración de ese sprint (generalmente 15 o 30 días). Al finalizar cada sprint, se valora el trabajo realizado y se prioriza los retrasos producidos, de haberlos. También se componen de reuniones diarias de corta duración (daily) para un análisis rápido del trabajo realizado el día anterior y las dificultades encontradas.
    - Otros: Desarrollo Adaptativo de Software (DAS), Método de Desarrollo de Sistemas Dinámicos (MDSD), Cristal, Desarrollo Impulsado por las Características (DIC), Desarrollo Esbelto de Software (DES), Modelado Ágil (MA), Proceso Unificado Ágil (PUA).
- Modelo orientado a la reutilización: este modelo está centrado en al reutilización, es decir, el proceso de desarrollo busca modelar el proyecto en base a que en un futuro sea posible reutilizar el código, los modelos, las tecnologías…

MOOC

En septiembre del 2008 George Siemens (Athabasca University) y Stephen Downes (National Research Council) crearon y publicaron el curso conocido como "CCK08: connectivism and connective knowledge" (Conectivismo y conocimiento Conectivo) al que Dave Comier (University of Prince Edward Island) calificó de "MOOC" acuñando por primera vez el termino. Actualmente, los MOOC han llegado

a ser un modelo utilizado y ofrecido por la mayoría de las universidades a nivel mundial.

La diferencia entre un MOOC y curso en línea habitual es que este último es un entorno cerrado, al que para acceder es necesario algún tipo de pago o matriculación previa, el número de participantes suele estar limitado; y sigue un modelo más convencional de enseñanza, mientras que en los MOOC se busca el protagonismo del alumno.

Los MOOC han demostrado ser muy útiles y fácilmente adaptables a disciplinas individuales como la programación, pero actividades grupales y desarrollos en equipo son aún un misterio a dilucidar, y realmente el número de MOOC en Ingeniería del Software o disciplinas similares es aún reducido en comparación con otras áreas de la computación.

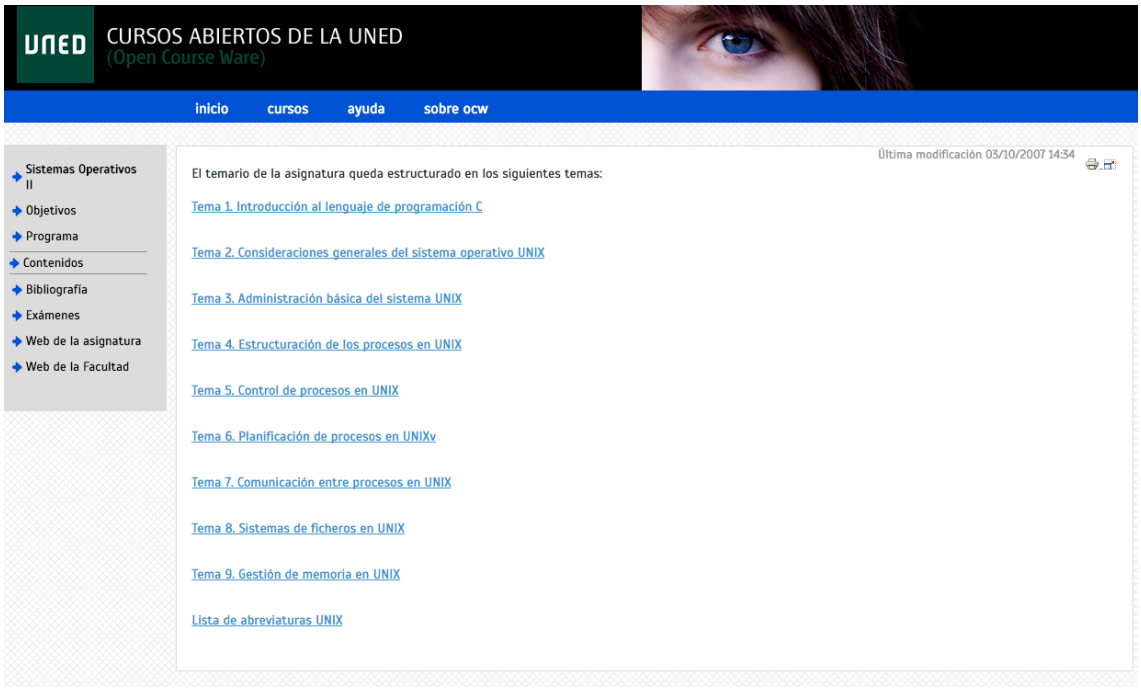
La principal dificultad radica en las actividades más "sociales" o que requieren una mayor interacción entre personas del mismo equipo o de distintos equipos. Actividades tales como la obtención de requisitos, técnicas de revisión o habilidades de presentación son las más difíciles de llevar a ser evaluadas por un MOOC. La necesidad de compartir un espacio o de ser habilidades más subjetivas de valorar (No es como un programa, que devuelve un resultado correcto o no) son las principales dificultades.

Los MOOC poseen una serie de características que los hacen de gran utilidad para investigadores y desarrolladores de software por lo siguiente:

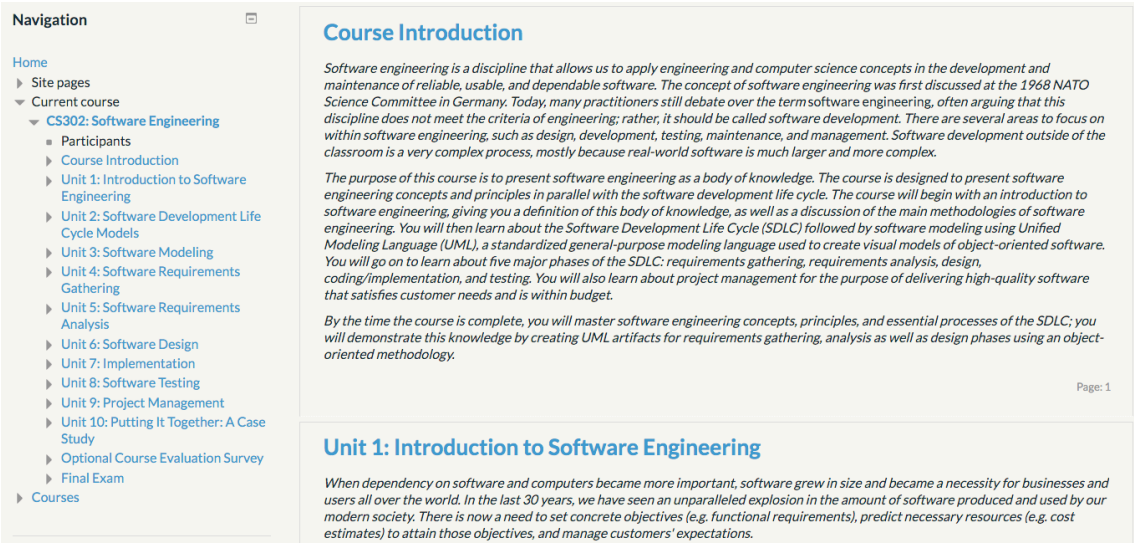
- Las herramientas y los métodos propios de la ingeniería del software son necesarios en el desarrollo óptimo de los MOOC, desde el punto de vista de la evaluación de los estudiantes, el desarrollo de un entorno gráfico que permita un aprendizaje adecuado.
- Los MOOC son, a su vez, útiles para profesionales o docentes desde el punto de vista de refrescar conocimientos o incluso aprender nuevo contenido para usarlo en sus clases o en su negocio.

Las principales plataformas que ofertan MOOC en Internet son Coursera, Udacity, EdX y Miríada X. Cada una de estas plataformas reúne una serie MOOC ofertados por universidades, entidades docentes o empresas; y cada una decide según su política si el acceso a estos cursos es gratuito o no, si ofrecen algún tipo de certificado o título al finalizarlos, etc. A estas hay que sumarles las plataformas concretas de las distintas universidades que también ofrecen MOOC.

Ejemplo de contenido de un MOOC



Ejemplo de contenido de un MOOC sobre la Ingeniería del Software



Hay diferentes estudios sobre la catalogación de los MOOC, sobre los resultados obtenidos de estos o sobre cómo medir la calidad de un MOOC. “Presente y futuro de los Massive Open Online Courses (MOOC): Análisis de la oferta completa de cursos de las plataformas Coursera, EdX, Miríada X y Udacity”, de Pablo Ruíz Martín , es un estudio bastante completo sobre este ámbito. En este documento se analizan las principales plataformas contenedoras de MOOC (Coursera, EdX, Miríada X y Udacity). Además, tras hablar sobre el origen de los MOOC, se establecen dos tipos de MOOC: los cMOOC, con una aproximación conectivista (que defiende una enseñanza colaborativa entre todos los miembros del curso), y los xMOOC, con una aproximación conductista (estilo más clásico, donde el profesor dirige el curso y enseña el conocimiento unidireccionalmente al alumno).

Un punto interesante citado en el artículo "Towards a Supercollaborative Software Engineering MOOC" es que la tasa de abandono de los MOOC es aproximadamente del 90 %, es decir, la mayoría de los alumnos que se matriculan o comienzan un MOOC no lo terminan. Esto es un aspecto negativo a tener en cuenta y que puede afectar a la viabilidad de un MOOC a la hora de tener en consideración su creación.

El artículo "Evaluación de la calidad pedagógica de los MOOC", trata establecer una evaluación de los distintos MOOC disponibles. Tras este estudio, lo que trata de clarificar el documento es que los MOOC poseen un gran potencial pedagógico, pero todavía les falta un último salto cualitativo para ser relevantes y se sugiere que deben dejar atrás los modelos clásicos del e-learning para buscar su propia identidad.

Para el futuro de los MOOC, el artículo "From MOOCs to SPOCs" destaca que hay quién afirma que los MOOC podrían cambiar de manera totalmente radical la formación académica en el futuro, valorando los aspectos positivos y negativos de esta posibilidad. A raíz de esto, el autor habla del concepto de "SPOC" (Pequeño curso en línea privada) que busca ser un complemento a la enseñanza en vez de un sustituto directo.

Concretando en los MOOC sobre Ingeniería del Software, no hay muchos estudios específicos en esta área. Se puede destacar el trabajo "Software engineering education (SEEd): Is software engineering ready for MOOCs?". Este estudio habla sobre el caso del MOOC "Software Engineering for Software as a Service (SaaS)" (actualmente conocido como "Agile Development Using Ruby on Rails"), un MOOC que se analizará en más profundidad más adelante. Lo destacable de este artículo es que denota la dificultad de enseñar las prácticas más "sociables" de la ingeniería del software, ya que, como dice el propio texto, evaluar un programa es relativamente sencillo (ya que se puede ejecutar y probar), pero evaluar relaciones o la aplicación de metodologías es más compleja y subjetiva.

Otro artículo destacable en este ámbito es "The past present and future of moocs and their relevance to software engineering", enunciando que los MOOC poseen un gran potencial para la comunidad de investigación y desarrollo de Ingeniería del Software. Por ejemplo, se podría usar los MOOC para ayudar a refrescar los conocimientos de los profesionales, o ayudar a la integración de nuevos profesionales a un desarrollo ya comenzado para que pueden aprender el uso de las herramientas, las metodologías…

Un artículo antes citado, "Towards a Supercollaborative Software Engineering MOOC", se centra en explicar un MOOC de Ingeniería del Software en un curso que trata de intentar simular de una manera lo más realista posible un entorno de desarrollo de un proyecto colaborativo a gran escala. Este enfoque de aproximación a casos reales se ve imprescindible en los MOOC sobre ingeniería del software del futuro.

MOOC SOBRE INGENIERÍA DEL SOFTWARE

A continuación se va a consultar todos los MOOC orientados a la informática en general y a la Ingeniería del Software en particular de las principales plataformas que ofertan MOOC.

Coursera

En primer lugar, se va a analizar la plataforma de Coursera. Coursera es una plataforma que recoge MOOC de diferentes universidades a lo largo del mundo y las organiza y agrupa por categorías y subcategorías con el objetivo de que los MOOC sean fáciles de encontrar y acceder.

Coursera dispone de una sección de MOOC, destinados a Ciencias de la Computación, en los que aborda el Desarrollo del Software, el Desarrollo Web y Móvil, Algoritmos, Seguridad Informática y de Redes, y Diseño y Producto.

En busca de MOOC que puedan estar relacionados con Ingeniería del Software, en análisis se centrará en los MOOC de Desarrollo del Software, Desarrollo Web y Móvil, y Diseño y Producto.

Coursera cruza los MOOC según si pueden corresponderse a varios de estos criterios. Por ejemplo, alguno de los MOOC ofertados son “Desarrollo y Diseño de Videojuegos” o “Diseño y Creación de videojuegos”, que están incluidos en las categorías de Desarrollo del Software y Patrones y Producto. O también

“Programando con Java para aplicaciones Android” que se incluye en las categorías de “Desarrollo Web y Móvil” y “Diseño y Producto”. Por tanto, se puede establecer que ambos estos cursos incluyen características de ambas categorías, es decir, se centran en la docencia sobre el lenguaje de programación, el entorno, etc.; pero también incluyen temario sobre diseño y producto, ámbitos que pertenecen a la Ingeniería del Software.

Se va a proceder a analizar estos MOOC y los MOOC propios de la categoría “Diseño y Producto” en busca del contenido del temario que sea más orientado a la Ingeniería del Software.

Desarrollo y Diseño de Videojuegos

El primer MOOC a analizar es “Desarrollo y Diseño de Videojuegos”. Este MOOC es impartido por la Universidad de los Andes (Colombia). Este MOOC se centra en el desarrollo de videojuegos con Unity y es un MOOC de pago. Este MOOC primero se centra en la explicación teórica de los videojuegos y luego aborda la explicación del lenguaje de programación y de las particularidades de programar en Unity. Pero el módulo interesante desde el punto de vista de la Ingeniería del Software es el dedicado a las metodologías ágiles para el desarrollo de videojuegos. En esta sección del curso se busca que el estudiante logre una compresión básica sobre cómo desarrollar tu videojuego siguiente una metodología de desarrollo ágil. Particularmente se centra en la metodología SCRUM y en sus elementos de planificación como:

- “Sprint backlog”. Un Sprint son las distintas iteraciones (bloques de tiempo cortos y fijos) en las que se divide el desarrollo a lo largo del tiempo. El “backlog” es la lista de las distintas tareas que se tienen que completar en esa iteración.
- Planificación de la entrega.
- Métodos de monitoreo. Estos métodos se componen de reuniones diarias (Daily) y de diagramas “burndown”, que consisten en gráficos que muestran el trabajo pendiente de la iteración.
- Métodos de retrospectiva. Se lleva a cabo mediante reuniones que se realizan al finalizar cada sprint donde cada miembro del equipo dan a conocer su impresión de la iteración.

Tabla resumen del MOOC

Plataforma	Coursera
Enlace web de acceso al MOOC	https://es.coursera.org/specializations/desarrollo-videojuegos
Organización	Dividido en cursos. Dentro de cada curso, el temario se divide por semanas.
Evaluación	Proyecto final al terminar todos los cursos
Temas	Introducción a desarrollo en Unity, Introducción al desarrollo de videojuegos, emprendimiento y métodos ágiles en el desarrollo de videojuegos, e introducción al desarrollo de videojuegos 3D en Unity.

Profundidad	Se profundiza en SCRUM como metodología ágil.

Gestión de Producto de Software

El siguiente MOOC analizado en esta plataforma es "Gestión de Producto de Software" de la Universidad de Alberta (Estados Unidos). El objetivo de este MOOC es que alumno aprenda a dominar las prácticas ágiles para el desarrollo de software, a dirigir un equipo de desarrollo y a interactuar con los clientes.

El MOOC está dividido en cursos planificados semana a semana, para acabar con un proyecto final que sirve como evaluación. Este curso no es gratuito, aunque proporciona un certificado al finalizarlo.

- El primer curso se centra en la importancia de la gestión de proyectos, definir el rol de un gerente de software, y sobre todo, en cómo las metodologías ágiles pueden mejorar el desarrollo y la calidad final de un producto software.
- El segundo curso continúa con la última parte del curso anterior, las metodologías ágiles y los procesos del desarrollo de software. Profundiza en varios procesos para el desarrollo del software, como el modelo en cascada o el Modelo Unificado. Cubre a su vez también los fundamentos de las dos principales metodologías ágiles existentes, SCRUM y Extremme Programming (Programación Extrema). El curso termina introduciéndose en algunos modelos más complejos como Lean y Kanban.
- En el tercer curso, se aborda las técnicas para la toma de requisitos, cómo realizar requisitos claros, visualizando las necesidades del cliente, maximizando la eficacia de la iteración con el cliente; y cómo adaptarse a los cambios en los requisitos. Se comienza definiendo lo qué es un requisito, y sus distintos tipos, para más adelante utilizarlos para definir los casos de uso o las historias de usuarios.
- El siguiente curso se centra en la planificación ágil, en cómo planificar de manera eficaz un desarrollo de software, crear y distribuir tareas, evaluar y planificar riesgos, generar estimaciones…
- El último curso abarca las técnicas para el seguimiento de proyectos, aplicar técnicas para medir y visualizar el progreso de un desarrollo e integrar prácticas de revisión ágiles.

Tabla resumen del MOOC

Plataforma	Coursera
Enlace web de acceso al MOOC	https://es.coursera.org/specializations/product-management
Organización	Dividido en cursos. Dentro de cada curso, el temario se divide por semanas.
Evaluación	Proyecto final al terminar todos los cursos
Temas	Introducción al Software de gestión de producto, procesos software y prácticas ágiles, necesidades del cliente y requerimientos software, planificación ágil, revisiones y parámetros para

	mejoras del software.
Profundidad	Especializado en metodologías ágiles. Abarca también temas de toma de requisitos.

Desarrollo de aplicaciones móviles con Android

A continuación, se sigue el análisis con el MOOC de "Desarrollo de aplicaciones móviles con Android", impartido por la Universidad Nacional Autónoma de México. Este MOOC es un curso de introducción a la programación en Android. Desde el punto de vista de la Ingeniería del Software lo más destacable del contenido son los patrones de diseño de Android, centrándose en el patrón de arquitectura de MVC (Modelo-Vista-Controlador) y MPV (Modelo-Vista-Presentador); el concepto de "Material Design", una normativa de diseño específica de Android, o la importancia de la realización de maquetas o "mockups" que guiarán el desarrollo de la aplicación.

Tabla resumen del MOOC

Plataforma	Coursera
Enlace web de acceso al MOOC	https://es.coursera.org/specializations/programacion-android
Organización	El temario se divide en semanas.
Evaluación	Proyecto final al terminar todos los cursos.
Temas	Básicos de Java, Programación orientada a objetos, lenguaje adicional de Java.
Profundidad	Algo de profundidad en el diseño.

Drive to Value with Agile Methods

Este MOOC trata sobre las metodologías de desarrollo ágil y los principios que las guían. Este MOOC ha sido creado por la Universidad de Virginia y está dividido en cuatro cursos y se evalúa mediante un proyecto final. Este MOOC tampoco es gratuito, pero si se finaliza correctamente este MOOC se obtiene un certificado. Cada curso en el que está divido consiste en:

- El primer curso es una introducción a las metodologías ágiles. Se centra en explicar los conceptos y prácticas que caracterizan una metodología ágil, y el concepto de historias de usuarios.
- El segundo curso se centra en el concepto de "Sprint", como diseñarlos y planificarlos.
- El tercer curso se enfoca más en la gestión del equipo en un proyecto ágil, en como adaptar el equipo de desarrollo a una metodología ágil.
- El cuarto y último curso abarca el tema de la motivación de adaptar una metodología ágil; y sobre todo el tema de la reutilización y las pruebas (como diseñar e integrar las pruebas en el flujo de desarrollo).

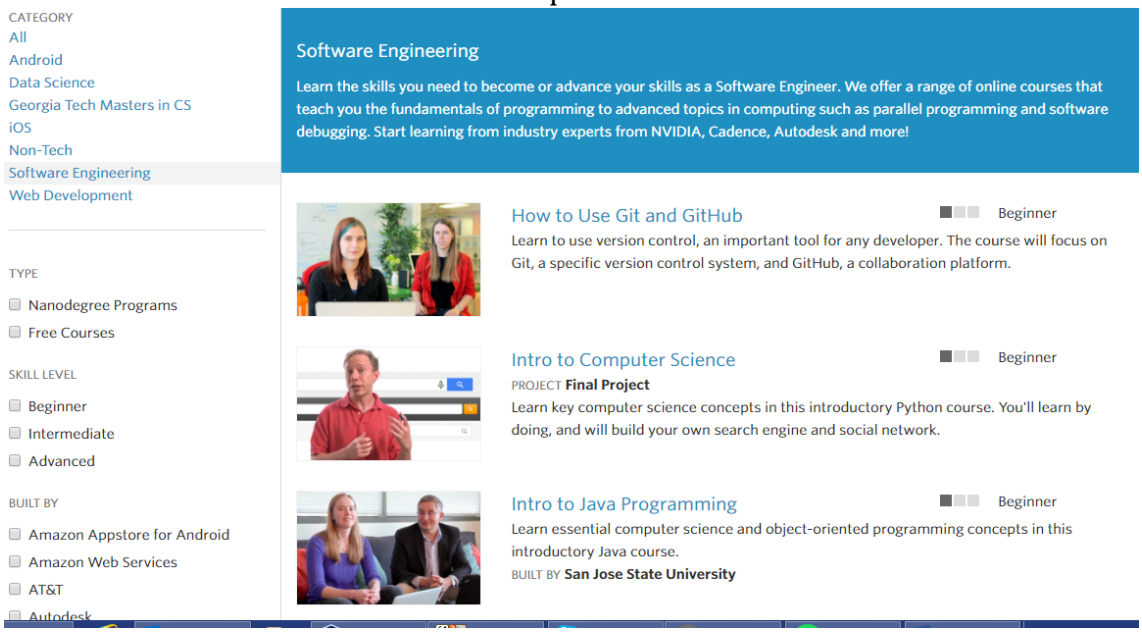
Tabla resumen del MOOC

Plataforma	Coursera

Enlace web de acceso al MOOC	https://www.coursera.org/specializations/agile-development
Organización	El temario se divide en semanas.
Evaluación	Proyecto final al terminar todos los cursos.
Temas	Introducción a la agilidad, diseño de sprints, gestionar un equipo ágil, experimentos con Agile.
Profundidad	Centrado es las metodologías de desarrollo ágiles.

Udacity

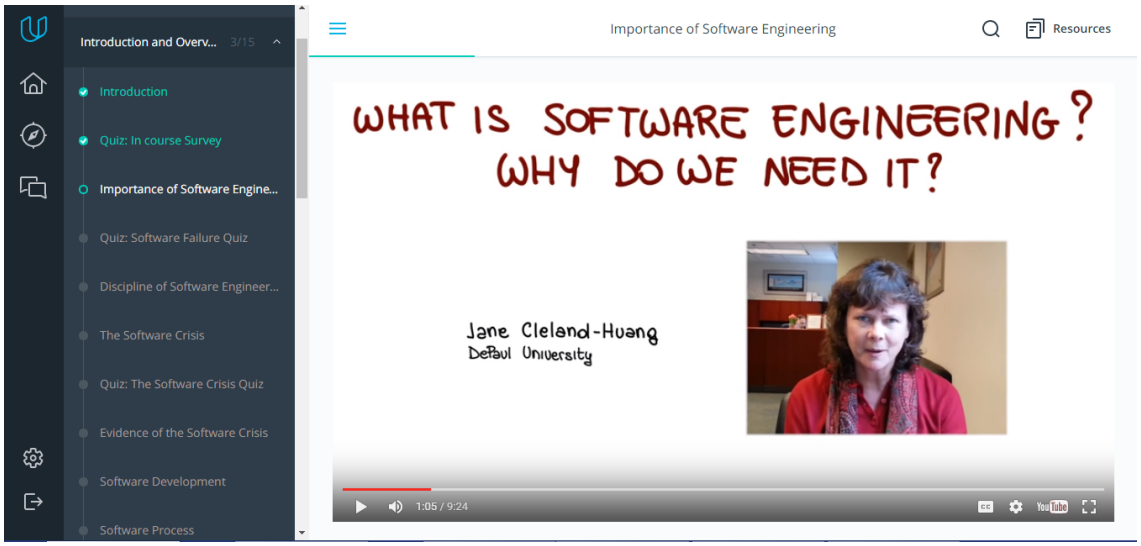
A continuación, se procede a continuar el análisis con la segunda plataforma de MOOC, Udacity, donde todos los cursos ofrecidos están relacionados con la informática. Udacity dispone de un apartado específico para la ingeniería del software con cursos ordenados además por dificultad.



En esta sección de la plataforma se incluyen tanto cursos de programación que incluyen temario de Ingeniería del Software como MOOC que se orientan más a ella. De este último grupo se encuentras MOOC como “Software Development Process” o “Software Testing”, que van a ser analizados a continuación.

Software Development Process

El MOOC “Software Development Process” es impartido por el Georgia Institute of Technology (Instituto de Tecnología de Georgia, Estados Unidos) y es de acceso gratuito. El objetivo de este MOOC es que el estudiante aprenda a elegir e implementar el mejor proceso de desarrollo de software. Las principales preguntas que intenta responder este MOOC son qué es la ingeniería del software y porqué es necesaria, preguntas que a lo largo del curso serán respondidas por expertos tanto del ámbito académico como del ámbito empresarial. El curso intercala vídeos explicativos emitidos en Youtube con preguntas tipo test y texto explicativo.



Durante todo el curso se hace hincapié en la importancia de una buena gestión del desarrollo más allá de la programación en sí, con el fin de construir un buen software minimizando los riesgos y errores en el proceso. Para ello, el curso aúna contenido sobre metodologías, técnicas y herramientas de desarrollo.

- Comenzando el curso, se trata de explicar el origen de la Ingeniería del Software a partir de los hechos históricos que dieron lugar a la necesidad de su existencia. Se establece la Crisis del Software de los años 60 como punto de partida para la creación de la Ingeniería del Software. El temario analiza las causas de esta necesidad, que el incremento de la complejidad demandada a los productos, la poca productividad de los programadores, la demanda creciente de software… Todos estos factores hicieron que fuera vital cambiar la manera de desarrollar el software.
- A continuación, el curso comienza con temario acerca de los procesos para el desarrollo del software (o modelos de ciclos de vida del software). Primero, se empieza por analizar sus puntos en común, para más tarde entrar en sus diferencias.
- Partiendo de los puntos en común, esta parte del curso comienza con las etapas básicas de un desarrollo de software: toma de requisitos, diseño, implementación, verificación y mantenimiento. Con los requisitos, el curso destaca la importancia de una buena toma de requisitos, y que un fallo, aún pequeño, en la toma de requisitos puede desatar fallos más graves conforme se avance en las etapas del desarrollo. En la parte del diseño, se explica de una manera esquematizada el diseño tanto del producto final como de los medios para desarrollar el producto. Con la implementación, simplemente se destacan cuatro principios básicos para una obtener una buena implementación (Reducción de la complejidad, anticipación a los cambios, facilitar la validación, uso de estándares). Para el apartado de verificación y validación, se explican estos conceptos y los distintos tipos de prueba (pruebas unitarias, de integridad o pruebas de sistema). Se finaliza con el mantenimiento, donde se explican conceptos como cambios en el

entorno, reporte de errores y peticiones de mejora. Más avanzado el curso se profundiza más en cada etapa.

- A continuación, el curso comienza con los distintos procesos de desarrollo del software y las diferencias entre ellos, y seguido, comienza con la explicación de cada uno de los procesos citados previamente de manera individual cada uno. Comienza con el proceso en cascada, continuando con el proceso en espiral, el proceso de prototipos, el Proceso Unificado de Rational y el proceso ágil, explicando cada una de sus etapas; destacando sus puntos fuertes y sus puntos débiles en comparación con los demás, y detallando cómo se debe elegir el proceso más adecuado para cada desarrollo. También hay apartados para los documentos usuales utilizados en el ciclo de vida del software y los errores más habituales en los desarrollos de software.
- Lo siguiente que aborda el MOOC son las herramientas de desarrollo. Empieza con el IDE (entorno de desarrollo) y se pone como ejemplo y se explica Eclipse con IDE para desarrollar, y conceptos básicos para trabajar con él, como los plugins, crear un proyecto, etc. La siguiente herramienta son los sistemas de control de versiones. Se explica de una manera general los sistemas de control de versiones y se concreta con el ejemplo de Git/Github y una guía de cómo utilizarlo.
- El siguiente paso del curso es profundizar en cada una de las etapas antes citadas para el ciclo de vida del desarrollo del software. Comienza con la especificación de requisitos, para la que se explica lo qué es, la importancia de una buena especificación de requisitos para el resto de etapas del desarrollo y cómo realizar una buena especificación. Detalla las diferencias entre requisitos funcionales y no funcionales, o requisitos del usuario y del sistema; técnicas para tomar y modelar requisitos y como priorizar los requisitos.
- A continuación, el curso explica el concepto de orientación a objetos y el lenguaje de modelado UML (Unified Modeling Language). Se analiza la orientación a objetos, destacando sus beneficios, y se explican conceptos relacionados con ella como las clases y los objetos. En lo que respecta a UML, se explica detalladamente como modelar un diagrama mediante UML, explicando los componentes de un diagrama de clases (como las clases, los atributos, las relaciones, etc.). Al igual que con los diagramas UML, el tema también abarca los diagramas de casos de uso.
- El temario del curso continúa con la arquitectura del software, explicando lo qué es, los elementos que componen una arquitectura, cómo diseñar una arquitectura software y detallando los tipos de estilos de arquitectura que existen.
- A continuación, el curso aborda el tema del diseño y los patrones de diseño. Los patrones de diseño aparecen divididos en Factory Method Patterns (patrones de método de fabricación), patrones de estrategia, y otros patrones (como patrón Decorador, patrón Observador, etc.). Finaliza con

unas indicaciones acerca de cómo elegir el mejor patrón para cada desarrollo.

- El siguiente tema del curso está centrado en RUP (Rational Unified Process/Proceso Unificado de Rational). Comienza con un poco de información sobre la historia y el origen de RUP, y va avanzado con una explicación detallada sobre RUP, centrándose y explicando cada uno de los aspectos que caracterizan RUP, como el desarrollo dirigido por los casos de uso, centrado en la arquitectura, interactivo e incremental; y las fases que lo componen (inicio, elaboración, construcción y transición).
- El punto siguiente de curso es la verificación y la validación, por tanto, esta parte se centrará en las pruebas. Se empieza por definir la diferencia entre fallo, defecto y error, para seguir con los distintos enfoques de verificación (pruebas, verificación estática, inspecciones de software, y comprobaciones formales), comparando las ventajas e inconvenientes de cada enfoque. Esta sección termina centrándose en las pruebas, explicando los distintos tipos de pruebas, como las pruebas alfa, las pruebas beta, las prueba de caja blanca y las pruebas de caja negra (dedicando dos temas aparte para estas dos últimas, temas que se van a analizar a continuación).
- Como se ha especificado anteriormente, el curso dedica una parte concreta del temario para las pruebas de caja negra y las de caja blanca, comenzando con las de caja negra. En ambas partes se explican lo que son las pruebas de caja negra y de caja blanca, y cómo realizar cada una de ellas.
- Continuando con el temario, el curso aborda el tema de las metodologías ágiles. Explica las ventajas de aplicar una metodología ágil a un desarrollo, y trata dos ejemplos de metodologías conocidas, como SCRUM o XP (eXtreme Programming), detallando alguna de sus características. De XP, por ejemplo, se explica la existencia de unos valores asociados a la metodología (simplicidad, comunicación, retroalimentación, valentía y respeto), y unas características concretas (integración continua, programación en parejas, diseño simple, etc.). En menos profundidad, se procede con una introducción a SCRUM para finalizar este tema.
- El último tema del curso es la refactorización del software, explicando este concepto, analizando la necesidad de la refactorización y los tipos de refactorización que existen, eligiendo alguno de estos para explicarlos. Concretamente, elige y explica colapsa jerarquías, fusionar condicionales, descomponer condicionales, extraer métodos y extraer clases. Tras esto, el temario expone cuando se debe refactorizar y cuando no, los riesgos asociados a ello.

Las evaluaciones en este MOOC consisten en una serie de test a lo largo del curso, tras acabar cada sección de temario en la que está dividido el curso. Aunque realmente no existe calificación, ya que no acredita ningún título al finalizarlo.

Este MOOC incluye temario sobre toma de requerimientos, arquitectura, diseño, pruebas, mantenimiento y calidad del software en general. Este MOOC es

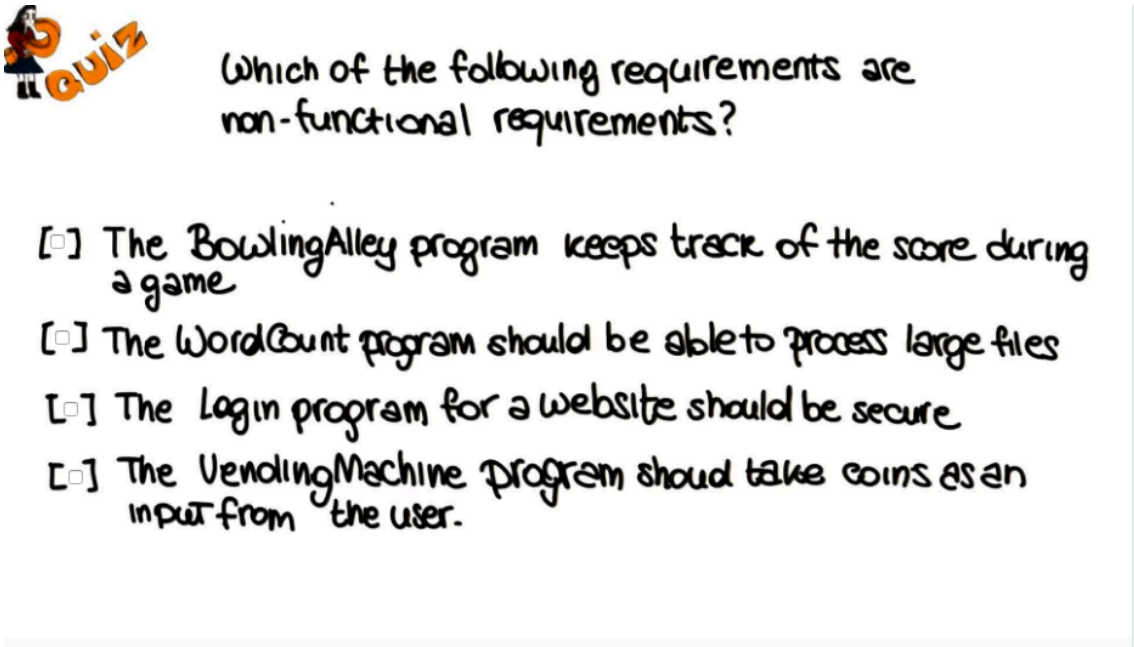
interesante desde el punto de vista de la Ingeniería del Software porque se centra más en el proceso de desarrollo del software que en la programación en sí.

Tabla resumen del MOOC

Plataforma	Udacity
Enlace web de acceso al MOOC	https://www.udacity.com/course/software-development-process--ud805
Organización	Dividido en lecciones. Cada lección a su vez, está dividida en módulos.
Evaluación	Test de autoevaluación.
Temas	Introducción a la Ingeniería del Software, modelos de ciclo de vida, entornos de programación, sistemas de control de versiones, toma de requisitos, software orientado a objetos, UML, arquitectura del software, análisis y diseño, patrones de diseño, Proceso Unificado de Desarrollo del Software, verificación y validación, pruebas de caja negra y de caja blanca, metodologías ágiles, refactorización del software.
Profundidad	Gran profundidad en todos los ámbitos de la Ingeniería del Software

Ejemplo de material docente

A continuación se puede ver un ejercicio de test sobre los requisitos no funcionales. Los test que se encuentran a lo largo de este curso son de este estilo.



Software Testing

El siguiente MOOC a analizar es “Software Testing”. Este MOOC se centra en las pruebas del software. También impartido por el Georgia Institute of Technology y de acceso gratuito, abarca tanto pruebas manuales como pruebas automáticas. En primer lugar se centra en las partes de un programa más sensibles a fallos y como realizar pruebas de la manera más óptima posible. Después explica los conceptos de pruebas automáticas y como se deben realizar y por último termina con diferentes formas de gestionar los errores encontrados en el software, reportarlos y tratarlos.

Al igual que el MOOC anterior, este dispone de vídeos explicativos durante el curso, pero en vez de test, el curso evalúa mediante la implementación de código en el momento, normalmente definiendo pruebas. El curso comienza con una aproximación más genérica sobre lo que son las pruebas de software.

Esta primera sección del curso comienza con cómo crear software testeable y con el concepto de aserciones, qué son, cuándo se usan y porqué utilizar aserciones. Continúa explicando los distintos tipos de prueba, como las pruebas unitarias, de integridad, de sistema y otras (pruebas de estrés, pruebas diferenciales y pruebas aleatorias). El curso también ofrece una serie de ejemplos de casos reales para denotar la importancia de las pruebas.

A continuación incluye un caso práctico para explicar las pruebas de caja negra, a través de un vídeo. Tras el vídeo, el curso te pide programar una prueba como evaluación.

El curso sigue con el concepto de “Coverage Testing” (Cobertura de código). En esta sección del curso, se trata de dilucidad la cantidad de pruebas que son necesarias para asegurar la calidad del software, es decir, la cantidad de código que ha sido cubierto por las pruebas, y maneras de medir esa cantidad. Durante este tema, se explican diferentes técnicas para intentar asegurar que la máxima cantidad código ha sido cubierta, como arboles biselados. Todo este temario mezclado también con ejemplos de casos reales. También se trata cuándo no merece la pena la cobertura, o cuándo directamente no funciona, y termina con un ejemplo de prueba de caja blanca automatizada.

Tras un par de ejercicios de codificar pruebas, el siguiente tema trata sobre las pruebas aleatorias, lo qué son, como definirlas y cómo usarlas sobre diferentes entornos o programas. Incluye varios ejemplos dónde las pruebas aleatorias son útiles (como la generación de un número de tarjeta de crédito), o motivos por los que las pruebas aleatorias pueden ser problemáticas. A continuación, profundiza en algunos conceptos relacionados con las pruebas aleatorias, como “Fuzz Testing” (técnicas de pruebas semi-aleatorias y automáticas), la generación de datos aleatorios, y el concepto de “Oráculo” (herramientas automáticas para determinar si el resultado de un test ha sido correcto o no), explicándolo y categorizándolo en tipos.

En el siguiente tema, se sigue con las pruebas aleatorias, pero desde un punto de vista práctico, es decir, cómo son las pruebas aleatorias en un ámbito real. A lo largo del tema, se van analizando cómo aplicar pruebas aleatorias a distintos casos en entornos y situaciones diferentes.

El último tema es similar al anterior, pero tratando cualquier tipo de prueba, es decir, analiza el uso de las pruebas en general en casos reales. Profundiza en los conceptos de optimización de los casos de prueba, y en el reporte de errores.

Tras cada tema, se incluyen ejercicios prácticos de pruebas es lo que hay que programar o modificar un código.

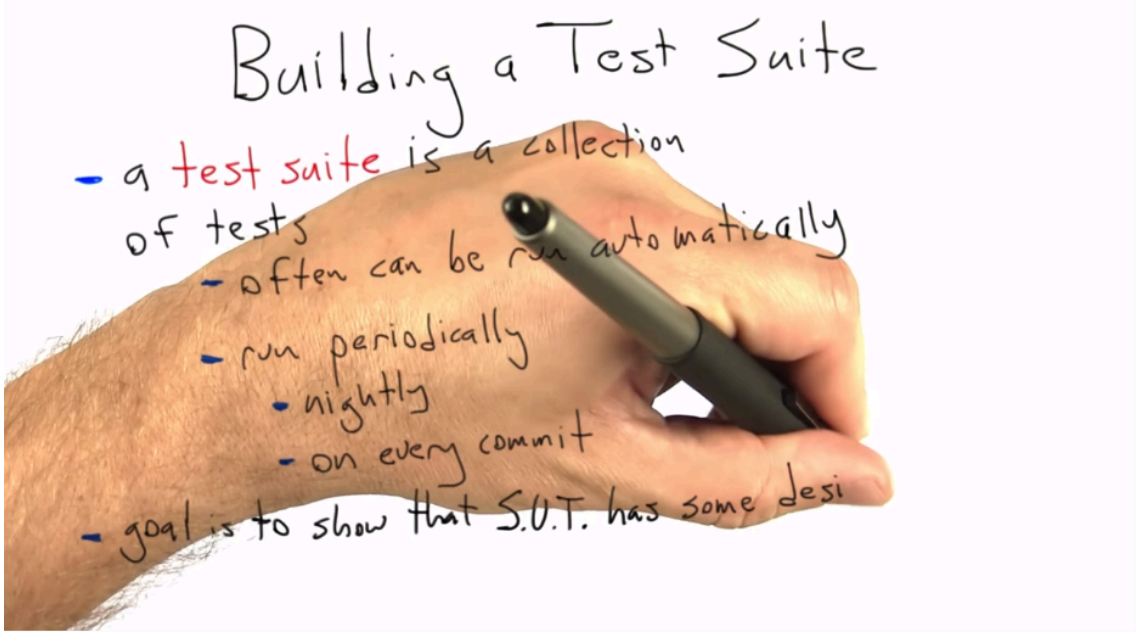
Para evaluar el curso, hay un examen al final de él. El examen consta de cuatro pruebas similares a los ejercicios intercalados durante el curso, uno por cada tema principal del temario. Aunque este curso, al igual que el anterior, no proporciona nada similar a una certificación o título.

Tabla resumen del MOOC

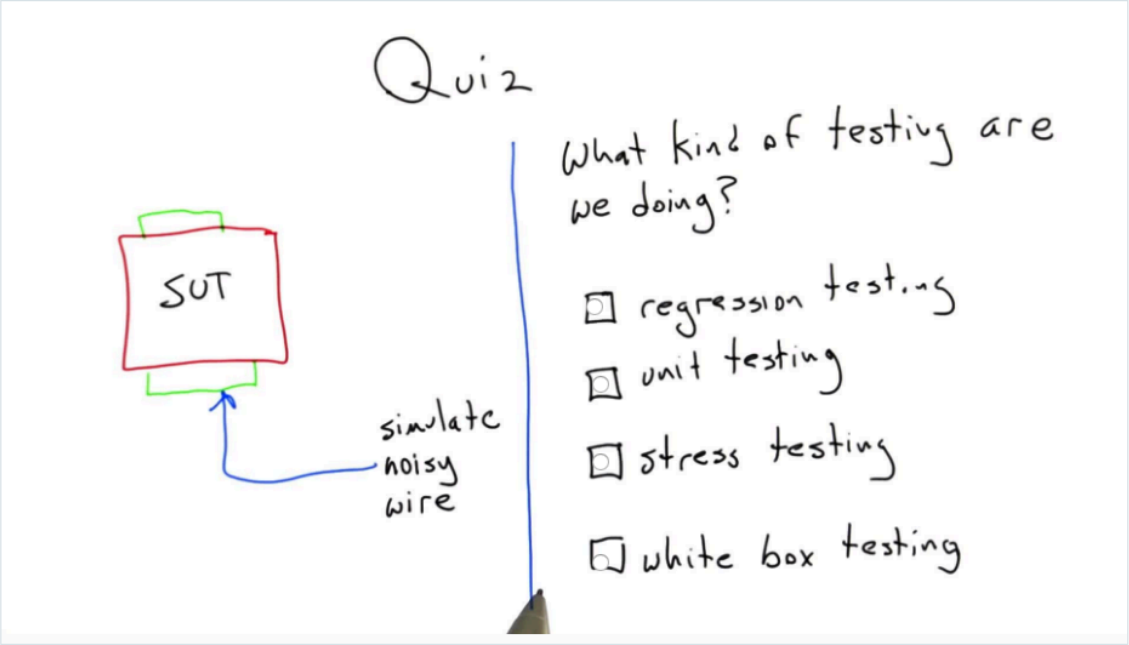
Plataforma	Udacity
Enlace web de acceso al MOOC	https://www.udacity.com/course/software-testing--cs258
Organización	Dividido en lecciones. Cada lección a su vez, está dividida en módulos.
Evaluación	Prueba final práctica.
Temas	Dominio, rangos, oráculos, y tipos de pruebas; cobertura de código, pruebas aleatorias, pruebas aleatorias avanzadas, consecuencias de las pruebas.
Profundidad	Alta profundidad en las pruebas software.

Ejemplo de material docente

Ejemplo de vídeo explicativo



Ejemplo de un test de evaluación



Ejemplo de un ejercicio de programación de una prueba

```
    def dequeue(self):
        if self.size == 0:
            return None
        x = self.data[self.head]
        self.size -= 1
        self.head += 1
        if self.head == self.max:
            self.head = 0
        return x

    def checkRep(self):
        assert self.tail >= 0
        assert self.tail < self.max
        assert self.head >= 0
        assert self.head < self.max
        if self.tail > self.head:
            assert (self.tail-self.head) == self.size
        if self.tail < self.head:
            assert (self.head-self.tail) == (self.max-self.size)
        if self.head == self.tail:
            assert (self.size==0) or (self.size==self.max)


# Provide full statement and parameter value coverage of the Queue class
def test():

```

Intro to Hadoop and MapReduce

Por último, en esta plataforma se podría destacar también el MOOC de "Intro to Hadoop and MapReduce", relacionado con programación distribuida y paralela. Desde el punto de vista de la Ingeniería del Software, se puede destacar la presencia en el temario de patrones de diseño para MapReduce.

Este curso gratuito ha sido realizado por Cloudera, empresa responsable de la distribución de Hadoop en colaboración con la propia plataforma de Udacity. La evaluación de este curso consiste en diferentes test entre cada tema dado para finalizar con la realización de un proyecto utilizando MapReduce.

Este curso comienza con una introducción al Big Data, para seguir profundizando en MapReduce, explicando cada una de las características de este framework. Concretamente, se va a analizar el último tema, relacionado con los patrones de diseño de MapReduce, que es el tema más relacionado con la Ingeniería del Software.

El tema de los patrones de diseño de MapReduce comienza con una visión general de estos patrones, especificando los distintos tipos, patrones de filtración, patrones de resumen, y patrones estructurales. Estas tres agrupaciones de patrones se explican más detalladamente conforme avance el tema. Tras explicar cada grupo de patrones, se realiza un test para evaluar los conocimientos.

Tabla resumen del MOOC

Plataforma	Udacity
Enlace web de acceso al MOOC	https://www.udacity.com/course/intro-to-hadoop-and-mapreduce--ud617
Organización	Dividido en lecciones. Cada lección a su vez, está dividida en módulos.
Evaluación	Test de autoevaluación.

Temas	Introducción a Big Data, introducción a HDFS, MapReduce y Hadoop; escribir programas en MapReduce, diseño de MapReduce
Profundidad	Profundiza en los patrones de diseño de MapReduce

Ejemplo de material docente

Ejemplo de pregunta con respuesta a desarrollar

Decision Process - Hadoop/MapReduce

Let's assume you have an active community site, similar to the Udacity forum, where users can post different information. You want to obtain some statistics about user behavior.

Is it a good idea to use MapReduce/Hadoop to process the data? Consider how each of the 3Vs of Big Data would affect this decision process.

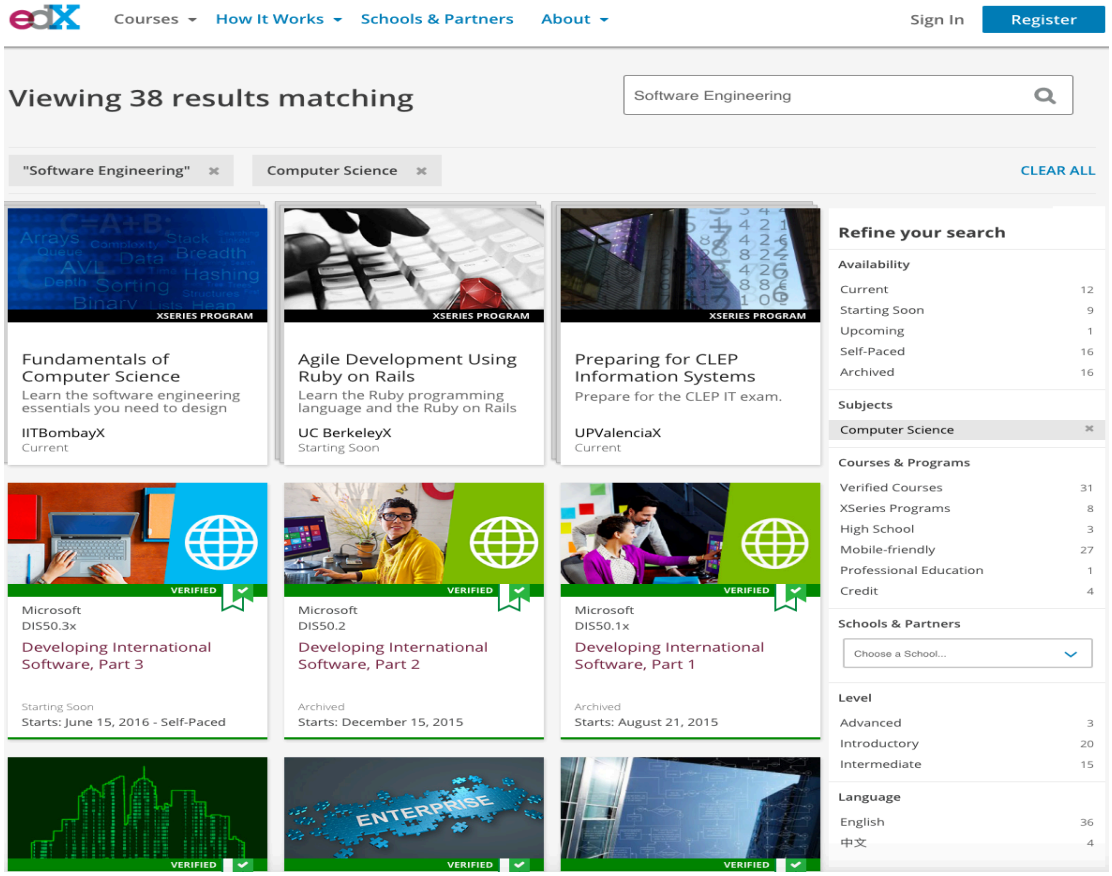
Ejemplo de ejercicio de programación

```
#!/usr/bin/python
import sys
import csv

# To run this code on the actual data, please download the additional dataset.
# You can find instructions in the course materials (wiki) and in the instructor notes.
# There are some things in this data file that are different from what you saw
# in Lesson 3. The dataset is more complicated and closer to what you might
# see in the real world. It was generated by exporting data from a SQL database.
#
# The data in at least one of the fields (the body field) can include newline
# characters, and all the fields are enclosed in double quotes. Therefore, we
# will need to process the data file in a way other than using split(","). To do this,
# we have provided sample code for using the csv module of Python. Each 'line'
# will be a list that contains each field in sequential order.
#
# In this exercise, we are interested in the field 'body' (which is the 5th field,
# line[4]). The objective is to count the number of forum nodes where 'body' either
# contains none of the three punctuation marks: period ('.'), exclamation point ('!'),
# question mark ('?'), or else 'body' contains exactly one such punctuation mark as the
# last character. There is no need to parse the HTML inside 'body'. Also, do not pay
# special attention to newline characters.

def mapper():
    reader = csv.reader(sys.stdin, delimiter='\t')
    writer = csv.writer(sys.stdout, delimiter='\t', quotechar='"', quoting=csv.QUOTE_ALL)
```

EdX



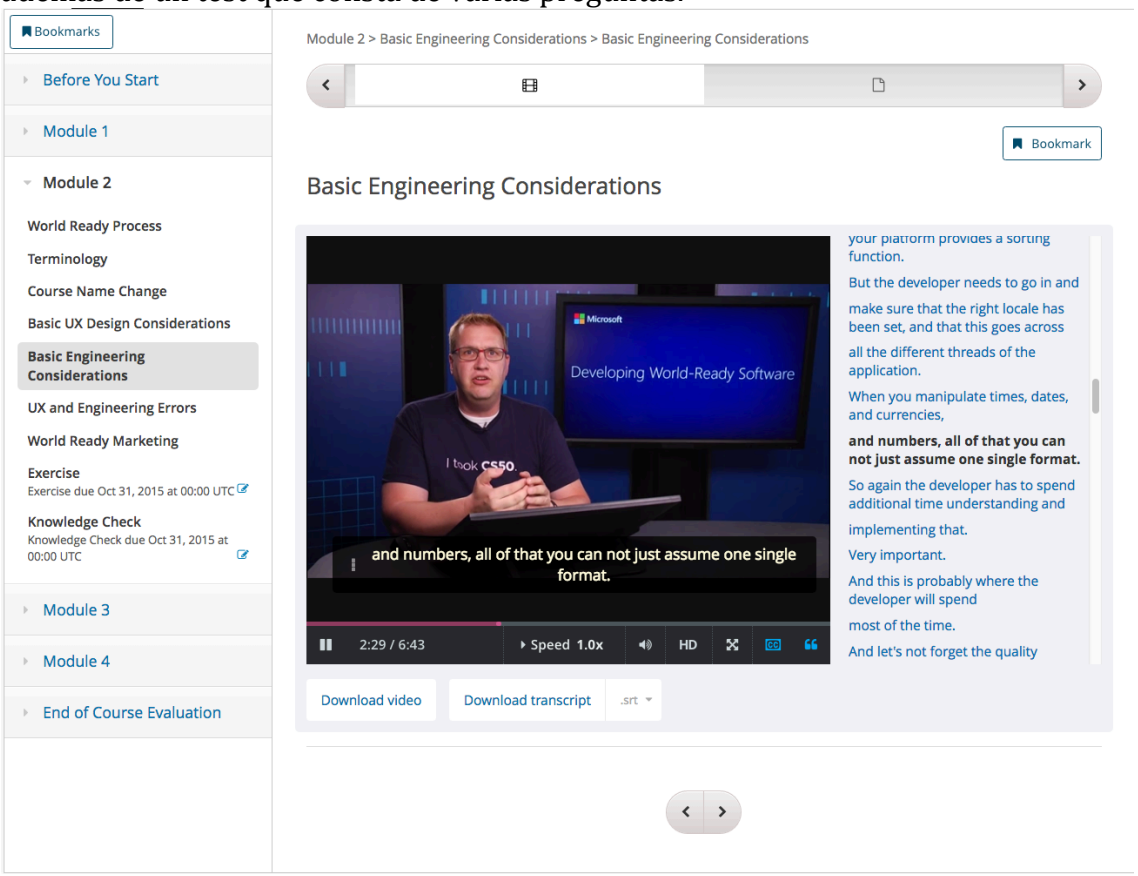
En la plataforma EdX se pueden encontrar una amplia gama de MOOC relacionados con la computación en general y con la ingeniería del software en particular. Un MOOC interesante encontrado en esta plataforma es el “Developing International Software”, un MOOC sobre cómo desarrollar software desde un punto de vista internacional, es decir, enfocar el desarrollo de software de una manera común en todo el mundo, siguiendo los mismos criterios y las mismas metodologías.

Developing International Software

Un MOOC interesante encontrado en esta plataforma es el “Developing International Software”, un MOOC sobre cómo desarrollar software desde un punto de vista internacional, es decir, enfocar el desarrollo de software de una manera común en todo el mundo, siguiendo los mismos criterios y las mismas metodologías. Este MOOC es impartido por Microsoft, es gratuito y se divide en tres partes, de las cuales las más interesantes desde el punto de vista de la Ingeniería del Software son las dos últimas.

Antes de comenzar el curso, la plataforma dispone de una introducción sobre cómo se va a evaluar el curso. Explica la distribución del curso en módulos, divididos a su vez en unidades didácticas, y cada unidad didáctica puede contener contenidos de

vídeo, material de lectura, problemas o ejercicios, y una serie de preguntas para evaluar los conocimientos aprendidos durante el curso. Algunas de las unidades, a su vez, pueden contener un foro de discusión para poder enviar preguntas o para la colaboración en general entre compañeros del curso. Pese a que las unidades están pensadas para seguirse en orden, el alumno puede saltar unidades de un lado a otro según crea conveniente. Para evaluar cada módulo, se pide al alumno un texto escrito en un documento Word que luego debe subir a la plataforma detallando un escenario concreto relacionado con el módulo recién finalizado, además de un test que consta de varias preguntas.



La primera parte del MOOC se centra en el porqué es importante una manera global de desarrollo del software y los recursos y metodologías necesarias para hacerlo posible.

- El primer módulo se centra en los pasos básicos para desarrollar un proyecto siguiendo esta filosofía de desarrollo global.
- En el segundo módulo de esta parte se explica la terminología básica y las consideraciones básicas que se tiene que seguir para diseñar el interfaz de usuario y para la manipulación y exposición de los datos, y los errores comunes asociados a ellos.
- El tercer módulo trata sobre el equipo y las herramientas necesarias para llevar a cabo un proyecto. Para formar un equipo en un ámbito de un desarrollo internacional, hay que tener en cuenta una serie de factores, para implicar a cada miembro del equipo en la importancia de tener en cuenta la internacionalidad del desarrollo y que el equipo esté comprometido con este objetivo, además de que un desarrollo global implica tener que tener en cuenta que se va a tratar con gente de distintas culturas. Desde el punto

de vista de las herramientas a utilizar, el curso hace hincapié en que dichas herramientas deben ser usadas para que el producto final sea igualmente utilizable en cualquier parte del mundo.
- El último módulo se centra más en el ámbito del mercado, trata de explicar cómo analizar el mercado y en cómo afecta ese análisis al desarrollo de un proyecto.

La segunda parte trata sobre las posibles diferencias culturales que pueden afectar a un desarrollo del software global. Factores como el lenguaje, el mercado de cada región, la cultura laboral, etc., son factores que obstaculizar esta meta. Para solventar estas dificultades, se centra en encontrar unos procesos de desarrollo y diseño efectivos en cualquier ámbito de desarrollo:

- El temario del primer módulo se centra en las consideraciones que hay que tener en cuenta a la hora de desarrollar un proyecto para que esté preparado para que sea distribuido en todo el mundo. Principalmente, el módulo se centra en el diseño de la interfaz de usuario, donde hay que tener en cuenta las diferencias del lenguaje, las diferencias culturales, y las diferencias entre países o regiones.
- El segundo módulo trata sobre el diseño centrado en las personas. Explica la importancia del usuario final del producto y de cómo hay que enfocar el diseño de la aplicación hacia ese usuario.
- El tercer módulo sigue al hilo del anterior, pero concreta el diseño orientado al usuario desde un punto de vista global, es decir, que ese usuario final puede ser de distintas partes del mundo y de cómo afecta eso al diseño de la aplicación.
- El cuarto y último módulo profundiza en las herramientas y técnicas de desarrollo empleadas en un desarrollo internacional. Explica el concepto de ejecutable mundial, qué es y sus beneficios. También explica el tratamiento del código que se debe seguir para tener un código multilenguaje e internacional, con algunas herramientas recomendadas para este objetivo.

La tercera parte consiste en, a partir de los detalles estudiados en las dos anteriores, enfrentarse a una serie de desafíos de desarrollo. Vuelve a hacer hincapié en el diseño de la interfaz de usuario (y las características que pueden afectar a un desarrollo internacional, por ejemplo, el lenguaje), en el tratamiento y envío de datos (diferentes países pueden tener diferentes regulaciones para el tratamiento de datos), y termina pasando por las prueba, el desarrollo y el mantenimiento (mantener y probar un software desarrollado en un entorno concreto desde otro entorno) de un código desarrollado de manera global. La segunda y tercera parte del curso se solapan, ya que según las recomendaciones de Microsoft, no es necesario haber realizado la segunda parte, aunque sí útil (Haber cursado la primera, sin embargo, si se considera muy recomendable, aunque tampoco es obligatorio). Esta parte del curso aún no está disponible para poder atenderla.

Tabla resumen del MOOC

Plataforma	EdX
Enlace web de acceso al MOOC	https://www.edx.org/course/developing-

	international-software-part-1-microsoft-dis50-1x
Organización	Cada parte del MOOC se divide en módulos.
Evaluación	Test de autoevaluación al final de cada módulo.
Temas	Introducción al Software globalizado, aspectos destacados, diseño efectivo, arquitectura, entorno de desarrollo, la interfaz de usuario, manipulación de datos, pruebas, desarrollo, mantenimiento,
Profundidad	Profundiza levemente en los aspectos de diseño, arquitectura, pruebas o mantenimiento.

Ejemplo de material docente

Ejemplo de enunciado de ejercicio propuesto en el curso. Una vez realizados, los ejercicios resueltos se publican en el foro del curso.

Exercise 3: Mapping technology

In Module Three, we have discussed what it means to make software localizable, as well as how to identify and address cultural sensitivities. This exercise will build on the work you did to create scenarios for Module Two's exercise. Now that you have a list of tasks and have broken at least one task into steps, it's time to explore what the User Experience will look and feel like.

ASSIGNMENT

Part One

1. Select a task from your list of tasks that takes at minimum two steps to complete. For each step of the task, please describe the following:
   - **What the user does:** List the steps that your software user will follow to complete the task, which you described for Exercise 2. You may need to update your descriptions, e.g., to explain in more detail what the user does.
   - **What the user sees, hears, or reads:** First, draw a rough sketch of what will appear in your software's User Interface as the user executes each step. Rather than try to upload the sketch to the edX website, you can submit a written description of it.
     - Is it possible that any content the user sees, hears, or reads may trigger cultural sensitivities?
     - Are there any other requirements to ensure that the User Experience will not only be functional but enjoyable?
   - **Technology considerations:** Make note of technologies required to enable each step. Be sure to highlight world-ready considerations.
2. Share 2 to 3 impressions on what you found interesting or challenging about this exercise.

Ejemplo de test para comprobación de conocimientos

Multiple Choice

(1 point possible)
What is the key to a successful globalization and localization process?

- Hiring engineers who specialize in globalization.
- Starting world-ready efforts at the beginning of the project.
- Adding comments to resource files that contain text strings.
- Releasing your software into multiple markets with an English UI as soon as possible, followed by fully localized versions as you complete them.
- Investing in research studies to understand international users.

?

SHOW ANSWER *You have used 0 of 1 submissions*

Agile Development Using Ruby on Rails

Otro MOOC interesante encontrado en esta plataforma es "Agile Development Using Ruby on Rails", antes conocido con el nombre de "Software Engineering for Software as a Service (SaaS)" de la Universidad de California, Berkeley creado por los profesores Armando Fox y David Patterson. Este MOOC está pensado como una introducción a la Ingeniería del Software, usando herramientas y metodologías de auténticos desarrolladores de Software. Este curso se centra, mediante el uso de Ruby On Rails, Cucumber y RSpec, en dar valor a las pruebas.

Mediantes este curso, se consiguen nociones importantes de control de versiones, SaaS (Software-as-a-Service/Software como servicio), patrones de diseño, test unitarios, Ruby on Rails o Cloud Computing.

Este MOOC se divide en dos partes, una primera parte calificada como básica y la segunda como avanzada. La primera parte se centra en abarcar todo un desarrollo de software sobre el framework de Ruby on Rails, incluyendo metodologías ágiles y técnicas de diseño, desarrollo y pruebas.

La segunda parte del MOOC se centra más en las relaciones entre modelos mediante el lenguaje Java y en los despliegues de la aplicación una vez terminada a los usuarios reales, y los problemas derivados de ello (rendimiento, persistencia de datos…). Desde el punto de vista de la Ingeniería del Software, lo más interesante de esta parte del curso es la aplicación de metodologías ágiles para mejorar y reutilizar código heredado, que según los creadores de este MOOC, es una tarea crítica en los programadores profesionales actuales. También se puede destacar otros puntos incluidos en el curso como la gestión de grupos de programación pequeños o los patrones de diseño (qué son y cuándo son interesantes aplicarlos).

Tabla resumen del MOOC

Plataforma	EdX
Enlace web de acceso al MOOC	https://www.edx.org/xseries/agile-development-using-ruby-rails
Organización	Cada parte del MOOC se divide en módulos.
Evaluación	-
Temas	Aplicar las técnicas de programación de Ruby on Rails al diseño, desarrollo y a las pruebas; desarrollo público en la nube de un software como servicio (SaaS), uso de Javascript para relaciones entre modelos; mejora, identificación y reparación de problemas, aplicar metodologías ágiles.
Profundidad	Profundiza en las metodologías ágiles aplicadas al framework de Ruby on Rails.

Computing: Art, Magic, Sciencie

También se puede destacar el MOOC "Computing: Art, Magic, Sciencie", impartido por el ETH Zurich (Escuela Politécnica Federal de Zurich). Este MOOC está divido también en dos partes. La primera no tiene un contenido demasiado interesante desde el punto de vista de la Ingeniería del Software, ya que es más una introducción de conceptos sobre computación y programación. En la segunda parte de este curso, sin embargo, sí se encuentran apartados más interesantes. Primero se enfoca más en técnicas y conceptos de programación, como la herencia o la concurrencia, pero conforme avanza el curso, aparecen temas relacionados con los patrones de diseño o una introducción a la ingeniería del software, donde se intenta destacar la importancia de ella conforme el desarrollo se va volviendo más complejo y ambicioso.

La introducción a la Ingeniería del Software la encontramos en el primer módulo de la segunda parte del curso. Esta introducción constituye una unidad del módulo. Durante esta unidad se van recorriendo todas las etapas de la Ingeniería del Software tras explicar su definición y destacar la importancia de la Ingeniería del Software para poder asegurar la calidad de un software. Con una serie de vídeos, se explica la toma de requisitos, el ciclo de vida del software y los procesos asociados a él, explicando algunos tipos de ciclos de vida, como el modelo en cascada y algunas de su variantes, el modelo en V, el modelo en espiral, el modelo incremental, los modelos ágiles y el modelo CMMI (Capability Maturity Model Integration/Integración de modelos de madurez de capacidades). Tras este módulo, se procede con un test para evaluar al alumno.

En el tercer módulo se encuentran los patrones de diseño, que se imparten en este módulo mediante una serie de test. Se centra en los patrones de diseño orientados a objetos, concretando en los patrones adaptadores, patrones de plantilla, patrones de estrategia, y patrones decoradores.

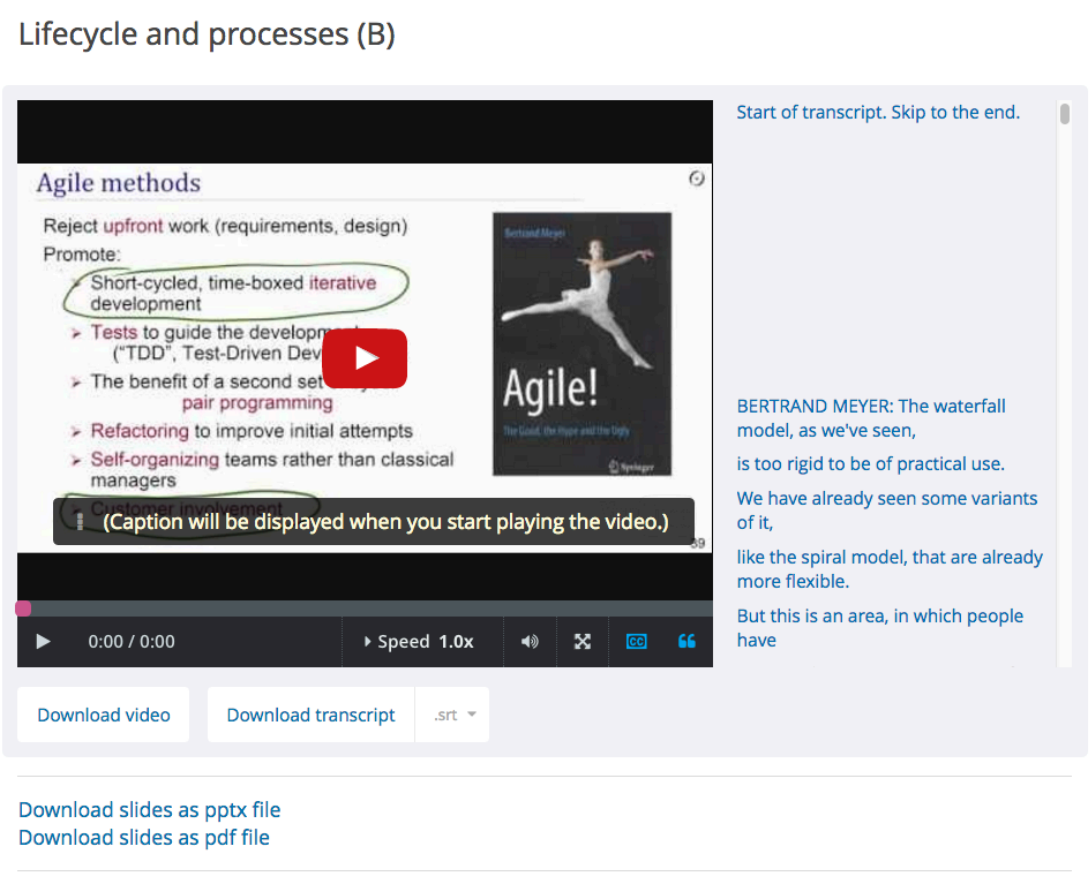
Para finalizar el curso, se realiza un examen final, que consiste en una serie de preguntas tipo test multirespuesta entre las que se encuentran preguntas sobre los módulos más relacionados con la Ingeniería del Software estudiados previamente.

Tabla resumen del MOOC

Plataforma	EdX
Enlace web de acceso al MOOC	https://www.edx.org/course/computing-art-magic-science-ethx-cams-2x#!
Organización	El temario se divide en semanas de curso.
Evaluación	Test autoevaluables, ejercicios de programación, y un examen final.
Temas	Introducción a la informática, conceptos principales de computación y programación, técnicas de programación orientada a objetos, la programación orientada a objetos "Eiffel", herencia múltiple, programación funcional, concurrencia, patrones de diseño, bucles, introducción a la ingeniería del software.
Profundidad	Profundiza en los patrones de diseño, pero no profundiza demasiado en la explicación de la ingeniería del software, en este tema es una introducción general a ella.

Ejemplo de material docente

Ejemplo de vídeo explicativo del temario, con la transcripción del guión del vídeo al lado



Ejemplo del examen de evaluación final

Question 6.1.Q1.1 Software Quality: external and internal factors

(1 point possible)
While external factors of software quality are visible to customers, internal factors are only visible to developers.

Which of the following can be considered internal factors of software quality?

- [ ] Good programming style
- [ ] Extendibility
- [ ] Information hiding
- [ ] Ease of use
- [ ] Timeliness

?

You have used 0 of 1 submissions

Question 6.1.Q1.2 Robustness

(1 point possible)
How would you define a Software System's Robustness?

- ○ The ability of the software system to perform reasonably according to the stakeholders.
- ○ The ability of the software system to perform according to specification, also in cases not covered by the specification.
- ○ The ability of the software system to perform according to specification, in cases covered by the specification.
- ○ The ability of the software system to perform "reasonably" in cases not covered by the specification.

?

You have used 0 of 1 submissions

Preparing for CLEP Information Systems Exam

Este MOOC impartido por la Universidad Politécnica de Valencia está específicamente creado como ayuda para preparar el examen CLEP (College Level Examination Program, un grupo de examen estandarizados creados por la "College Board"; una corporación americana sin ánimo de lucro con el objetivo de ampliar el acceso a educación de alto nivel) de Sistemas de Información.

El MOOC se divide en cinco cursos, siendo el tercer curso el interesante desde el punto de vista de la ingeniería del software, ya que esta parte trata sobre el desarrollo del software e incluye temario sobre metodologías de desarrollo y

modelos de ciclos de vida. El resto de partes del MOOC tratan desde una introducción general a las tecnologías de la información, pasando por los componentes hardware hasta llegar a la parte de desarrollo con un tema concreto de programación y finalizando con las implicaciones sociales del software. En la siguiente captura se puede apreciar esta división.

Courses in this XSeries Program

This is a series of 5 courses that can be taken in any order.

CLEP Information Systems and Computer Applications, Part 1: IT
Prepare for the CLEP Information Systems and Computer Applications exam. Part 1 will focus on IT in organizations.
Learn more

Self-Paced
Enroll Now
I would like to receive email from Universitat Politècnica de Valencia and learn about its other programs.

CLEP Information Systems and Computer Applications, Part 2: Hardware
Prepare for the CLEP Information Systems and Computer Applications exam. Part 2 will focus on hardware.
Learn more

Self-Paced
Enroll Now
I would like to receive email from Universitat Politècnica de Valencia and learn about its other programs.

CLEP Information Systems and Computer Applications, Part 3: Software Development
Prepare for the CLEP Information Systems and Computer Applications exam. Part 3 will focus on software development.
Learn more

Self-Paced
Enroll Now
I would like to receive email from Universitat Politècnica de Valencia and learn about its other programs.

CLEP Information Systems and Computer Applications, Part 4: Programming
Prepare for the CLEP Information Systems and Computer Applications exam. Part 4 will cover the programming elements of the exam.
Learn more

Self-Paced
Enroll Now
I would like to receive email from Universitat Politècnica de Valencia and learn about its other programs.

CLEP Information Systems and Computer Applications, Part 5: Social Implications
Prepare for the CLEP Information Systems and Computer Applications exam. Part 5 will cover the social implications of technology.
Learn more

Self-Paced
Enroll Now
I would like to receive email from Universitat Politècnica de Valencia and learn about its other programs.

Por tanto, se va a centrar el análisis en la tercera parte de desarrollo del software. Esta parte comienza con una introducción al desarrollo de software, comenzando con los conceptos básicos de este ámbito, con los modelos de ciclo de vida del software, las herramientas necesarias para optimizar un desarrollo. Básicamente el curso hace una introducción a los conceptos de la ingeniería del software, explicando las actividades que la componen y los tipos de modelos de ciclos vidas más comunes que existen.

Para la evaluación, al final de cada unidad en la que se divide cada parte del curso hay un examen además de test autoevaluables conforme se va avanzando en el temario.

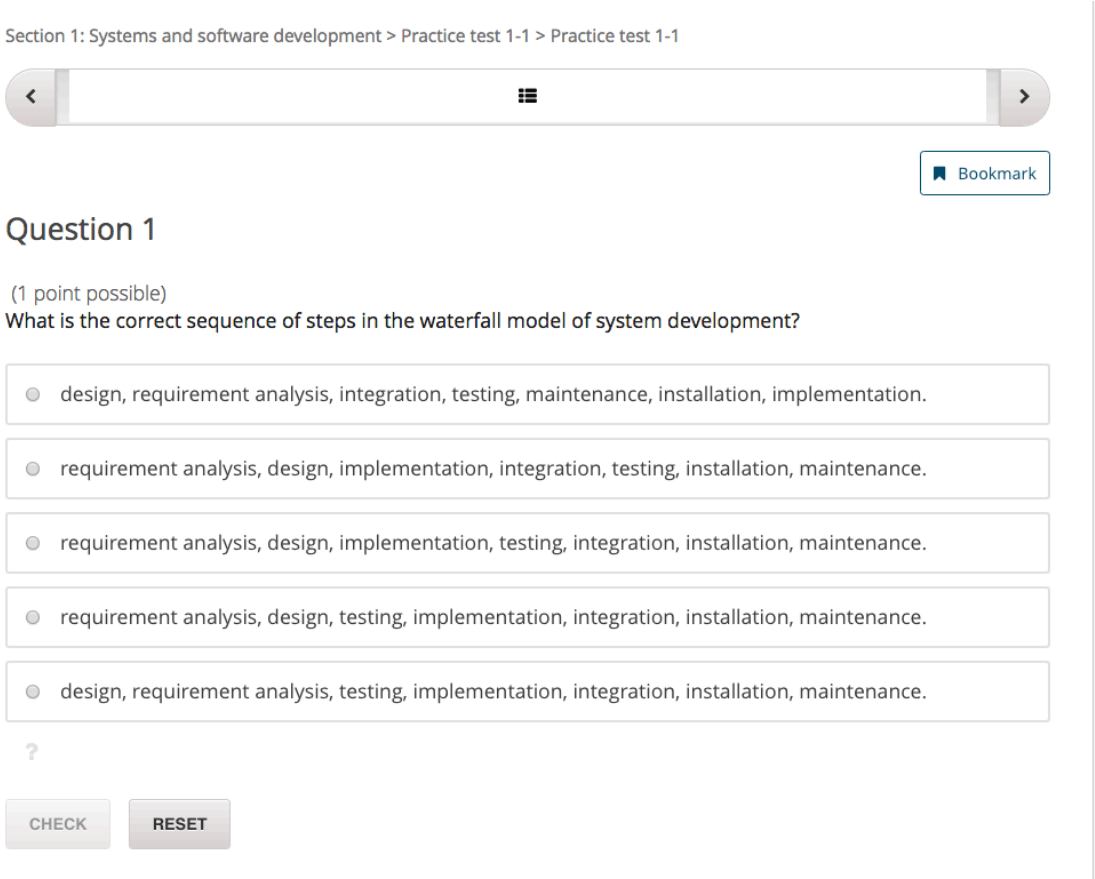


Tabla resumen del MOOC

Plataforma	EdX
Enlace web de acceso al MOOC	https://www.edx.org/xseries/preparing-clep-information-systems-exam
Organización	El temario se divide en semanas de curso.
Evaluación	Test autoevaluables, un examen tipo test al final de cada unidad y un examen final de cada parte del curso
Temas	Tecnologías de la información, hardware, desarrollo del software, programación, implicaciones sociales.
Profundidad	Introducción a los conceptos de Ingeniería del Software.

Enterprise Software Lifecycle Management

Este MOOC, impartido por la National Research Nuclear University "MEPhI" (Moscú, Rusia), trata sobre los modelos, las prácticas y los métodos de desarrollo de software. Este curso es una introducción a la gestión del ciclo de vida del software, aunque todavía está en construcción y su contenido no está accesible. El curso será gratuito, aunque si al finalizarlo se quiere obtener un certificado, este no lo es.

El curso está dividido en semanas, como se puede ver en la imagen.

Week 1: Software Engineering: From Chaos to Technology
Introduction to basic software engineering concepts, such as software system and software application, as well as the concept of enterprise.

Week 2: Software Lifecycles
Introduction to software development lifecycle, including its basic stages, entry/exit criteria and key deliverables.

Week 3: Basic Lifecycle Models
Overview of the software lifecycle models, including waterfall, incremental and rapid prototyping.

Week 4: Advanced Lifecycle Models
Comparative study of more complex software lifecycle models, including spiral and object-oriented.

Week 5: Methodologies: The Rational Process
Overview of the software development methodologies, including roles, artifacts and key deliverables for the Rational Unified Process (RUP).

Week 6: Methodologies: The Microsoft Process
Comparative study of the RUP methodology and Microsoft Solution Framework.

Week 7: Conclusion and Perspectives
Summary of the lessons learned on managing the lifecycles for large-scale software development, including major challenges and promising approaches.

Cada semana corresponde a un tema concreto. La primera semana es una introducción a los conceptos básicos de ingeniería del software, continuando en la segunda semana con las características básicas de los ciclos de vida de desarrollo de software, para continuar en la siguiente semana profundizado en los más básicos y simples (como el modelo en cascada, el incremental o el de prototipado rápido), continuando con los más avanzados (como el modelo en espiral o el orientado a objetos). En la quinta y sexta semana se profundiza en unos modelos concretos, el RUP (Rational Unified Process) y el proceso de Microsoft, acompañándolos con una comparación entre ellos. La última semana consiste en un conclusiones sobre los modelos tratados a lo largo del curso.

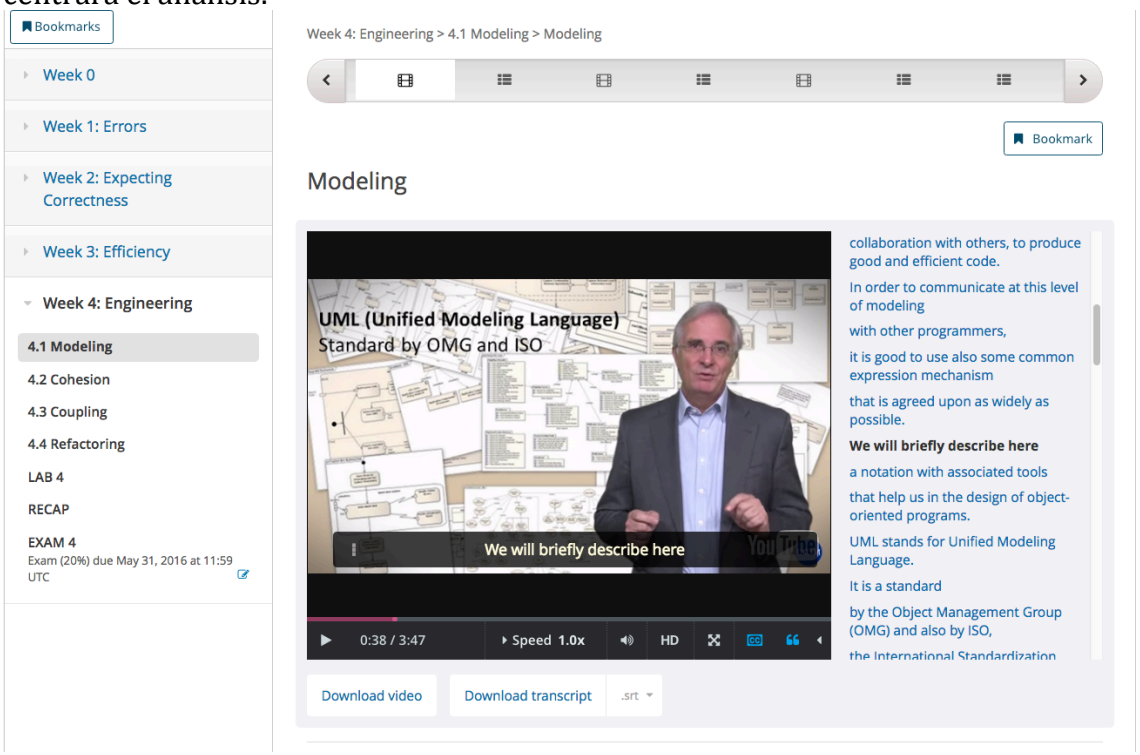
Tabla resumen del MOOC

Plataforma	EdX
Enlace web de acceso al MOOC	https://www.edx.org/course/enterprise-software-lifecycle-management-mephix-mephi001x
Organización	El temario se divide en semanas de curso.
Evaluación	Examen final
Temas	Introducción a la ingeniería del software, ciclos de vida del software, modelos básicos de ciclos de vida, modelos avanzados de ciclos de vida, el proceso de Rational, el proceso de Microsoft.

Profundidad	Principalmente, profundiza en los modelos de ciclo de vida

Introduction to Programming with Java Part 2: Writing Good Code

Este curso creado por la Universidad Carlos III (Madrid) trata de enseñar los fundamentos de la programación con Java. Este curso está dividido en dos partes: la primera parte está totalmente dedicada a la programación puesto que es una introducción a la principios básicos de la programación en general y a la programación orientada a objetos en particular. La segunda parte sigue centrada en la programación, pero en la última semana del curso se abordan temas relacionados con la ingeniería del software, por tanto esta será la sección donde se centrará el análisis.



Esta sección contiene una explicación de UML, su definición y cómo utilizarlo. Incluye, por ejemplo, el modelado de diagramas de clases.

Durante incluye una serie de test autoevaluables a lo largo del curso, con una ejercicio práctico y un examen final al finalizar cada semana de temario.

(2 points possible)

UML

Select the correct definitions and sentences:

- [ ] UML is a programming language that can be compiled
- [ ] UML is a modeling language which is intended to provide a standard way of visualizing the design of a system
- [ ] UML class diagrams represent classes and interfaces of a system
- [ ] You can use an Eclipse plug-in to generate UML class diagrams

?

Note: Make sure you select all of the correct options—there may be more than one!

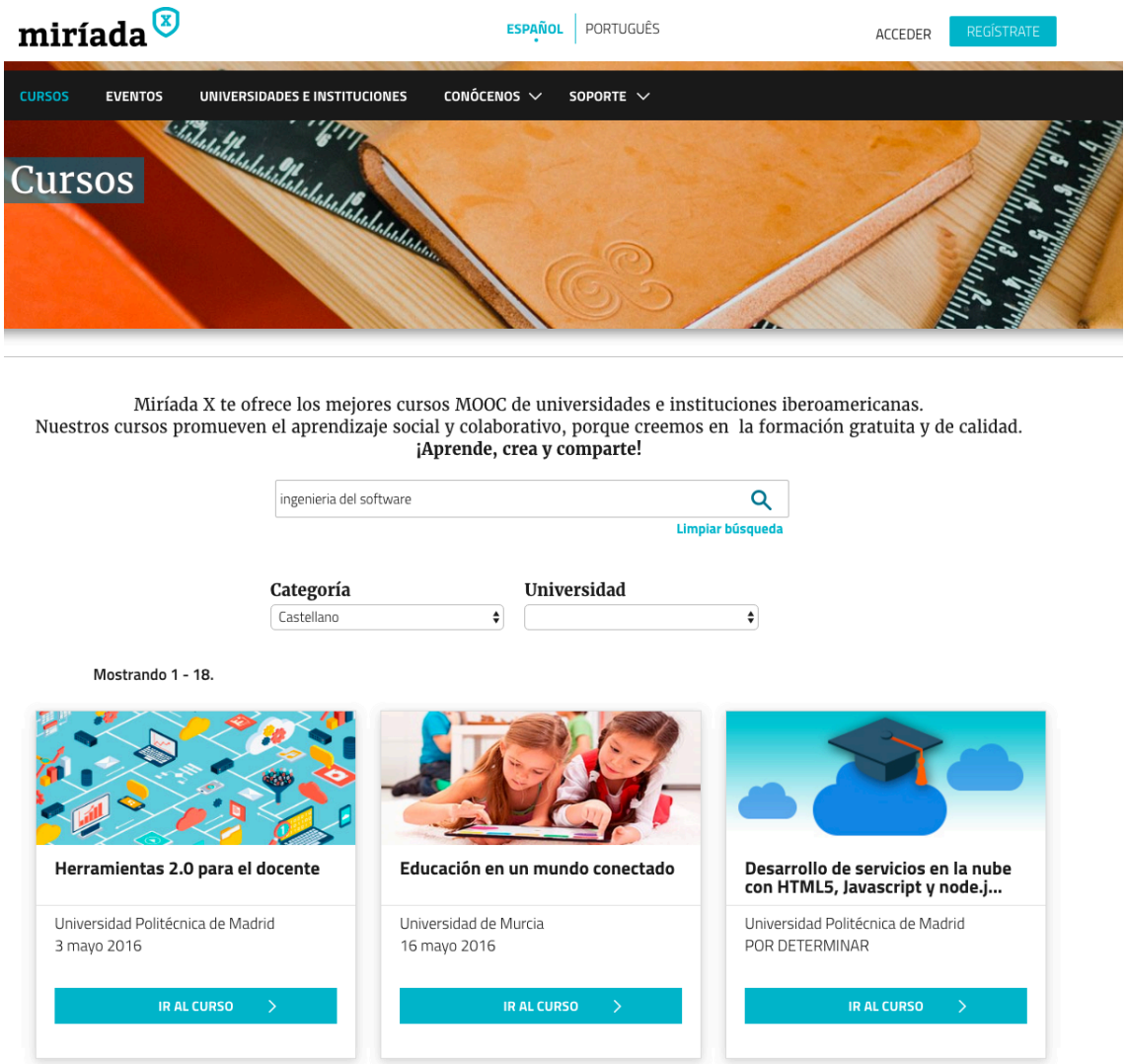
FINAL CHECK SAVE *You have used 0 of 1 submissions*

Tabla resumen del MOOC

Plataforma	EdX
Enlace web de acceso al MOOC	https://www.edx.org/course/introduction-programming-java-part-2-uc3mx-it-1-2x
Organización	El temario se divide en semanas de curso.
Evaluación	Test autoevaluables, ejercicios prácticos y examen final
Temas	Detectar y corregir errores, eficiencia, modelado, cuestiones éticas.
Profundidad	Se centra mucho más en la programación, y no profundiza demasiado en el modelado.

Miríada X

La última plataforma a analizar es Miríada X. En ella, se encuentran algunos MOOC interesantes desde el punto de vista de la Ingeniería del Software como “Agilidad y Lean. Gestionando los proyectos y negocios del s. XXI”.



Agilidad y Lean. Gestionando los proyectos y negocios del s. XXI

El MOOC de "Agilidad y Lean. Gestionando los proyectos y negocios del s. XXI" trata de abarcar todo el proyecto de desarrollo de software con metodologías actuales y modernas. Este MOOC es impartido por la Universidad Rey Juan Carlos de Madrid. El curso está divido en módulos de aprendizaje o lecciones y contiene temario sobre metodologías ágiles, testing y el concepto de Peopleware:

- En primer lugar, como lección 0, se establece una introducción sobre el curso. En esta introducción se explican los objetivos del curso, la temática que trata, la modalidad de evaluación (que consiste en un test cada vez que se termina una lección y un examen final después de finalizar el curso), el foro del curso con las normas de su uso (útil para consultar dudas), y algunas referencias que pueden ser de ayuda para el curso (libros y blogs).
- El primer módulo del curso explica la diferencia entre los ciclos de vida tradicionales (como el modelo en cascada) con los ágiles y las ventajas de estos últimos. Explica en más profundidad los ciclos de vida ágiles, tratando sus características y principios.

LECCIÓN 1: CONSTRUIR SOFTWARE NO ES COMO CONSTRUIR COCHES O CASAS

Ciclo de vida ágil

Sería un **ciclo de vida iterativo e incremental, con iteraciones cortas (semanas)** y **sin que dentro de cada iteración tenga porque haber fases lineales** (tipo cascada). A partir de la anterior, matizaciones, adaptaciones, etc., hay por cada metodología ágil que existe.

Quizá el caso más popular es el de **Scrum**. Hace ya sus años, en el 85, y en la primera presentación oficial de Scrum, Ken Schwaber, uno de sus creadores, hablaba sobre el ciclo de vida de Scrum, y sus diferencias con los anteriores ciclos de vida

Según comenta Ken Schwaber, **el ciclo de vida en Cascada y el Espiral** cierran el contexto y la entrega al inicio de un proyecto. **La principal diferencia entre cascada, espiral e iterativo y los ciclos de vida ágiles**, concretamente en Scrum, es que estos últimos asumen que el análisis, diseño, etc., de cada iteración o Sprint son impredecibles. **Los Sprints, o iteraciones cortas, no son (o a priori no tienen porqué) lineales y son flexibles**.

Pero, como comentaba, **cada metodología de las llamadas ágiles**, FDD, Crystal, DSDM, XP, etc., **matizará su ciclo de vida.**

- A continuación, el segundo módulo trata del "Peopleware", concepto que trata de establecer un tercer aspecto aparte del Software y el Hardware. El Peopleware se refiere a todo ámbito o detalle que tenga que ver con las personas involucradas en el desarrollo de software (formación del equipo, organización y gestión del personal…). Explica la relevancia de buscar y retener a los miembros del equipo más adecuados para el proyecto, la importancia del entorno (tanto físico como el software) de trabajo y porqué es mejor trabajar con equipos pequeños y multifuncionales.
- El tercer módulo trata sobre el concepto de "Product Owner" (persona con la visión exacta de qué se quiere desarrollar y que debe transmitirlo al resto del equipo), qué es y porque es vital para un desarrollo ágil. A continuación se tratan las historias de usuario y su relación con el Product Owner, como deben hacerse, que información contienen, y como interpretarlas correctamente.
- Se prosigue con el cuarto módulo, que se centra en Scrum y la explicación sobre su aplicación a un proyecto. Profundiza en los conceptos de "Product Backlog" y de "Sprint"; en las reuniones de planificación, los "daily" (reuniones diarias), y las reuniones de revisión, y las reuniones de retrospectiva. A continuación, se explican los gráficos "BurnDown" y en cómo interpretarlos para medir el progreso el proyecto. También se destacan los beneficios de Scrum frente a otras metodologías.
- La quinta lección explica lo qué es la planificación ágil, centrándose en la estimación. Se enseña la técnica de estimación "Planning Poker", y las "historias de usuario" que utiliza esta técnica. Destaca a continuación, los peligros de las estimaciones y cómo se pueden realizar estimaciones más realistas y precisas, aunque también se expone si realmente es posible conseguir una estimación lo suficientemente precisa o incluso si tiene sentido estimar.

LECCIÓN 5: LA PLANIFICACIÓN ÁGIL

La unidad de estimación: Puntos historia

Históricamente, la unidad clásica de estimación del "tamaño" de un requisito ha sido el **Punto Función**. Pero, en los últimos años, los **puntos historia** se han convertido en una unidad de tamaño muy usada, principalmente en proyectos ágiles. Por ejemplo, **la técnica de estimación Planning Poker, que explicaremos más adelante, emplea los Puntos Historia**.

Según la definición más popular, y que a la vez más confusión ha dado, un punto historia es una fusión de a) la cantidad de esfuerzo que supone desarrollar la historia de usuario (o elemento del Product Backlog, de manera general), b) la complejidad de su desarrollo y c) el riesgo inherente. Esta es la definición más popular, es la que originariamente escribió M. Cohn , uno de los precursores del término.

- En el sexto módulo se trata la metodología Lean, y la técnica Kanban. Comenzando con Lean, se explica su origen, las diferencias entre una metodología Lean y una metodología ágil, y como trata Lean los "desperdicios" en un proyecto. A continuación, se explica Kanban con una serie de ejemplos.
- El último modulo trata sobre el Testing en metodologías ágiles y el concepto de "Deuda técnica". Esta parte del MOOC se centra en las consecuencias del desarrollo, incluyendo el desarrollo y la realización de pruebas, o el concepto de "Deuda técnica", que se refiere a las consecuencias de un desarrollo apresurado o descuidado, que puede ser producido por una mala planificación o por una mala implementación de las metodologías estudiadas anteriormente. Incluye un apartado dedicado a Cucumber, y otros dedicados a la automatización de pruebas en móviles con Calabash.
- Para finalizar el curso, se procede a realizar un examen para evaluar al alumno.

Tabla resumen del MOOC

Plataforma	Miríada X
Enlace web de acceso al MOOC	https://miriadax.net/web/agilidad-y-lean-gestionando-los-proyectos-y-negocios-del-s-xxi-5-edicion-
Organización	Dividido en lecciones.
Evaluación	Test autoevaluables tras cada lección, y un examen final.
Temas	Introducción, construir software, peopleware, Product Owner e historias de usuario, SCRUM, planificación ágil, Lean y Kanban, y Deuda Técnica y Testing ágil.
Profundidad	Profundiza en metodologías ágiles y en las pruebas.

MOOC de otras plataformas

Software Engineering

En la plataforma "saylor.org" se puede encontrar el MOOC "CS302: Software Engineering". Este curso trata de abarcar todo el proceso y etapas de Ingeniería del Software. El curso está dividido en unidades teóricas que incluyen lecturas y vídeos recopilados de distintas universidades, páginas web, etc. Tras cada unidad hay un test de evaluación de conocimientos y al finalizar el curso hay un examen final. A continuación, se dispone un análisis de cada una de las unidades:

- El curso comienza con una introducción al curso, explicando el temario que se tratará en cada unidad del curso y los objetivos finales del curso.
- Tras la introducción al curso, lo próximo es una introducción a la Ingeniería del Software. El objetivo de esta unidad es explicar y definir la ingeniería del software, la diferencia de esta frente a la Informática, clasificarla según características o tipos, describir sus tres tipos principales, explicar códigos éticos y profesionales relacionados con ella y establecer las características que debe tener un buen software.
- La segunda unidad trata sobre los modelos de ciclo de vida del desarrollo del software. El temario define este término, y analiza dos tipos de ciclos de vida, el secuencial y el iterativo, estableciendo una comparación entre ambos.
- En la tercera unidad se aborda el tema del modelado de software, concretamente se centra en explicar UML, sus características y su aplicación.
- La cuarta unidad se centra en técnicas para la toma de requisitos, definición de tipos de datos, y definir los conceptos de solicitud de propuesta y elaboración de propuesta.
- En la quinta unidad se profundiza en los requisitos, en cómo definir buenos requisitos, establecerlos por funcionales y no funcionales y las actividades relacionadas con el análisis de requisitos. También se incluye temario acerca de metodologías orientadas a datos, orientadas a procesos y orientadas a objetos, con un análisis de cada una de estas.

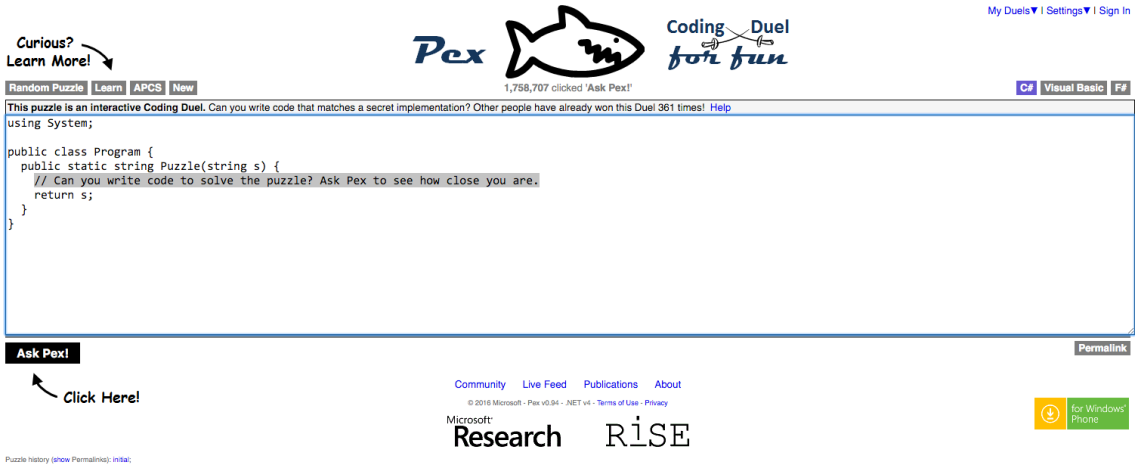
- A continuación, la sexta unidad trata el tema del diseño, explicando los principios del diseño de software, y las actividades principales de diseño. En el temario de esta sección se discute el diseño arquitectónico, es decir, la organización del sistema, descomposiciones modulares, controles de flujo, etc.
- La implementación es el tema principal de la séptima unidad. En ella se define lo qué es la programación, los diferentes lenguajes de programación y sus características, y la relación entre el hardware y el software y cómo elegirlos adecuadamente.
- En la octava unidad el curso trata el tema de las pruebas, definiendo su terminología, y explicando distintas estrategias de testing, incluyendo pruebas de caja blanca, pruebas de caja negra y pruebas top-down y bottom-up. También se explica cómo definir adecuadamente un pan de pruebas que incluya pruebas unitarias, pruebas de integración y sistemas para medir la cobertura de código alcanzada por las pruebas.
- La novena unidad trata sobre la gestión de proyectos, explicando el rol del gestor del proyecto, identificando sus áreas de responsabilidad. Explica los conceptos relacionados con la gestión de proyectos desde el punto de vista del propio proyecto en sí (planificación, programación, ejecución…), desde el punto de vista de las personas (contratación, motivación, evaluación…), y desde el punto de vista de la gestión de los cambios (configuración, aplicaciones, software…).
- Para finalizar el curso, la última unidad aúna todos los conceptos estudiados previamente en un caso de estudio. En este caso se pedirá tomar requisitos, modelar el software mediante UML, desarrollar un plan de pruebas, etc.

Tabla resumen del MOOC

Plataforma	Saylor.org
Enlace web de acceso al MOOC	https://learn.saylor.org/course/cs302
Organización	Dividido en unidades, que a su vez de dividen en subunidades de temario.
Evaluación	Test autoevaluable al final de cada unidad, y un examen final.
Temas	Introducción a la ingeniería del software, modelos de ciclo de vida del software, modelado del software, toma de requisitos, análisis de requisitos, diseño del software, implementación, pruebas de software.
Profundidad	Dentro de que abarca una gran parte de todo lo relacionado con la ingeniería del software, no ofrece una gran profundidad dentro de cada tema.

Pex4Fun

Otro MOOC analizado por su particularidad es Pex4Fun lanzado en 2010 por Microsoft, cuyo objetivo es enseñar programación e ingeniería del software por medio del aprendizaje interactivo basado en juegos.



Por medio de Pex4Fun, los profesores pueden crear nuevas áreas virtuales con nuevos contenidos o modificar y personalizar los existentes. Pex4Fun es accesible desde cualquier navegador y dispone de un editor de código, y pretende animar al estudiante a seguir aprendiendo por medio de juegos y desafíos que podrá ir completando definiendo código. Aunque en los primeros niveles se orienta más a la programación en sí, para estudiantes más avanzados dispone de juegos más enfocados a la ingeniería del software como pruebas, análisis y resolución de problemas…

Tabla resumen del MOOC

Plataforma	Microsoft Research
Enlace web de acceso al MOOC	http://pex4fun.com/
Organización	Por niveles
Evaluación	Desafíos de programación.
Temas	No dispone de temario
Profundidad	No profundiza demasiado en temario.

RESUMEN Y CONCLUSIONES

A continuación mediante distintas tablas se resume los análisis realizados anteriormente. Como se ha explicado al principio de este texto, los MOOC sobre ingeniería del software analizados han sido divididos en tres grupos; el primer grupo se compone de los MOOC exclusivamente centrados en la programación y en los lenguajes de programación, el segundo grupo incluye los MOOC de programación que en alguna parte poseen temario sobre algún ámbito de la ingeniería del software (metodologías, pruebas, etc.), y el tercer y último grupo abarca los MOOC dedicados a la ingeniería del software totalmente.

En la siguiente tabla se resumen las plataformas analizadas en función de los MOOC que albergan.

	MOOC Totales	MOOC sobre Informática	MOOC sobre Ingeniería del Software	MOOC sobre Ingeniería del Software (Tipo 1)	MOOC sobre Ingeniería del Software (Tipo 2)	MOOC sobre Ingeniería del Software (Tipo 3)
Coursera	Más de 500	Más de 300	55	45	5	5
Udacity	Más de 100	Más de 100	20	17	1	2
EdX	Más de 900	Más de 200	29	20	7	2
Miríada X	Más de 100	21	7	6	0	1

En las tablas que aparecen a continuación, se establece una comparación entre los MOOC estudiados lo largo de este documento, tanto desde el punto de vista docente, como desde el punto de vista de contenido del temario.

	Desarrollo y Diseño de Videojuegos	Gestión de productos Software	Desarrollo de aplicaciones móviles con Android	Drive to Value with Agile Methods
Clasificación	2	3	2	3
Idioma	Español	Inglés	Español	Inglés
Tiempo de Estudio	-Primer curso: 8 semanas. -Segundo curso: 4 semanas, 4-6 horas por semana. -Tercer curso: 5 semanas, 3-5 horas por semana. -Cuarto curso: 5 semanas.	-Primer curso: 2 semanas, 1-2 horas por semana. -Segundo curso: 4 semanas. -Tercer curso: 4 semanas, 3-4 horas por semana. -Cuarto curso: 4 semanas, 1-2 horas por semana, 1-3 horas por semana.	-Primer curso: 3 semanas, 3-5 horas por semana. -Segundo curso: 5 semanas, 1-3 horas por semana. -Tercer curso: 5 semanas, 3-5 horas por semana.	-Primer curso: 5 semanas, 3-4 horas por semana. -Segundo curso: 5 semanas, 3-5 horas por semana. -Tercer curso: 4 semanas, 3-5 horas por semana. -Cuarto curso: No especificado.

		-Quinto curso, 6 semanas, 3-5 horas por semana.		
Vídeos	Sí	Sí	Sí	Sí
Foros	No	Sí	No	No
Chats	No	No	No	No
Tutorización	No	No	No	No
Coste	177 € el curso completo. Se puede pagar cada curso por separado y de manera individual.	341 € el curso completo. Se puede pagar cada curso por separado y de manera individual.	193 € el curso completo. Se puede pagar cada curso por separado y de manera individual.	425 € el curso entero. Se puede pagar cada curso por separado y de manera individual.
Certificación o diploma	Sí	Sí	Sí	Sí
Test de evaluación	Sí	Sí	Sí	Sí
Test de autoevaluación	Sí	Sí	Sí	Sí
Prácticas o ejercicios resueltos	No	No	No	Sí
Orden de estudio establecido y obligatorio	Sí	Sí	Sí	

	Software Develoment Process	Software Testing	Intro to Hadoop and MapReduce
Clasificación	3	3	2
Idioma	Inglés	Inglés	Inglés
Tiempo de Estudio	Tres meses aproximadamente (Suponiendo 6 horas por semana).	Un mes aproximadamente (Suponiendo 6 horas por semana).	Un mes aproximadamente (Suponiendo 6 horas por semana).
Vídeos	Sí	Sí	Sí
Foros	Sí	Sí	Sí
Chats	No	No	No
Tutorización	No	No	No
Coste	Gratis.	Gratis	Gratis
Certificación o	No	No	No

diploma			
Test de evaluación	No	No	No
Test de autoevaluación	Sí	Sí	Sí
Prácticas o ejercicios resueltos	Sí	Sí	Sí
Orden de estudio establecido y obligatorio	No	No	No

	Developing International Software	Software Engineering for Software as a Service (SaaS)	Computing: Art, Magic, Sciencie
Clasificación	3	3	2
Idioma	Inglés	Inglés	Inglés
Tiempo de Estudio	-Primer curso: 4 semanas, 3 horas por semana. -Segundo curso: 11 semanas, 3 horas por semana. -Tercer curso: 4 conjuntos de problemas, 5-8 horas cada uno	18 semanas por curso (2 cursos), 12 horas cada semana.	-Primer curso: 7 semanas, 5-10 horas por semana. -Segundo curso: 6 semanas, 5-10 horas por semana.
Vídeos	Sí	Sí	Sí
Foros	Sí	Sí	Sí
Chats	No	No	No
Tutorización	Sí	Sí	Sí
Coste	El curso es gratis. La certificación cuesta 49 $.	El curso es gratis. La certificación cuesta 99 $.	Gratis
Certificación o diploma	Sí	Sí	No
Test de evaluación	No	-	No
Test de autoevaluación	Sí	-	Sí
Prácticas o ejercicios resueltos	Sí	-	Sí
Orden de estudio establecido y obligatorio	No	No	No

	Preparing for CLEP Information Systems Exam	Enterprise Software Lifecycle Management	Introduction to Programming with Java Part 2: Writing Good Code
Clasificación	2	3	2
Idioma	Inglés	Inglés	Inglés
Tiempo de Estudio	18 semanas, 3 horas por semana.	7 semanas	5 semanas, 5-7 horas por semana.
Vídeos	Sí	Sí	Sí
Foros	Sí	Sí	Sí
Chats	No	No	No
Tutorización	No	No	Sí
Coste	El curso gratis. La certificación son 50 $.	El curso gratis. La certificación son 49 $.	El curso gratis. La certificación son 50 $.
Certificación o diploma	Sí	Sí	Sí
Test de evaluación	Sí	-	Sí
Test de autoevaluación	Sí	-	Sí
Prácticas o ejercicios resueltos	No	-	Sí
Orden de estudio establecido y obligatorio	No	No	No

	Agilidad y Lean. Gestionando los proyectos y negocios del s. XXI	Software Engineering	Pex4Fun
Clasificación	3	3	2
Idioma	Español	Inglés	Inglés
Tiempo de Estudio	7 semanas (50 horas estimadas)	135 horas.	-
Vídeos	Sí	Sí	No
Foros	Sí	No	No
Chats	No	No	No
Tutorización	No	No	No
Coste	Gratis.	El curso es gratuito. Acceder al examen final cuesta 25 $.	Gratis.
Certificación o diploma	Sí	Sí	No
Test de evaluación	Sí	Sí	No
Test de	Sí	Sí	No

autoevaluación			
Prácticas o ejercicios resueltos	No	No	Sí
Orden de estudio establecido y obligatorio	No	No	Sí

Como conclusión, se puede concretar que el principal desafío al que se enfrentan los creadores de MOOC relacionados con la Ingeniería del Software es el cómo abordar la docencia del ámbito social de los desarrollos, es decir, todo lo que tiene que ver con relaciones entre los miembros del grupo, gestión del equipo, relaciones con los clientes… Este ámbito es difícil de reproducir en curso online y es un reto que otros MOOC no tienen que tener en cuenta a la hora de enseñarlo y evaluarlo.

Este es uno de los puntos que hacen que los MOOC de Ingeniería del Software sean diferentes a otros MOOC del área de la Informática o la Computación como por ejemplo la programación, que es más fácil de evaluar ya que un código o programa es sencillo de determinar si está bien desarrollado o no, pero ejercicios relacionados, por ejemplo, con la toma de requisitos, son más difíciles de evaluar al tener ese componente social y subjetivo que requiere tratar con un usuario o cliente, analizar una necesidad, etc.

La plataforma que ha demostrado tener más MOOC relacionados en mayor o menor medida con la ingeniería del software ha sido EdX, aunque el acceso a muchos de ellos está restringido por tiempo, es decir, solo están disponibles en unas fechas concretas; y además tras acabar los cursos si se quiere obtener un certificado se debe pagar. Coursera también presenta un importante numero de MOOC de ingeniería del software, pero la principal desventaja de Coursera es que todos sus cursos son de pago y no se puede ni acceder al contenido si se ha pagado la matrícula del curso.

Desde el punto de vista de la ingeniería del software, se podría decir que de todos los MOOC analizados el más completo es “Software Development Process” de la plataforma Udacity, tanto por cantidad de contenido como por profundidad en cada uno de los temas. Un defecto a destacar de este MOOC quizá sea que es principalmente teórico.

BIBLIOGRAFÍA

Ardis, M.A. & Henderson, P. B. (2012). Is Software Engineering Ready for MOOCs? - Software Engineering Education (SEEd). Recuperado de: http://personal.stevens.edu/~mardis/papers/SEEd1209.pdf

Ruíz Martín, P. (2013). Presente y futuro de los Massive Open Online Courses (MOOC): Análisis de la oferta completa de cursos de las plataformas Coursera, EdX, Miríada X y Udacity. Universidad Complutense de Madrid, Facultad de ciencias de

la documentación, España. Recuperado de:
http://eprints.ucm.es/23502/2/MOOCs.pdf

Bilingsley. W. & Steel, J. R. H. (2014). Towards a Supercollaborative Software Engineering MOOC. 36th International Conference on Software Engineering, Hyderabad, India. Recuperado de: https://dl.acm.org/citation.cfm?id=2591157

Roig, R. , Mengual-Andrés, S & Suárez, C. (2014). Evaluación de la calidad pedagógica de los MOOC. Recuperado de:
http://www.ugr.es/~recfpro/rev181ART2.pdf

Fox, A. (2013). From MOOCs to SPOCs. Recuperado de:
http://cacm.acm.org/magazines/2013/12/169931-from-moocs-to-spocs/abstract

Dasaranthy, B., Sullivan, K., Schmidt, D.C. Fisher, D.H., Porter, A. (2014). The Past, Present, and Future of MOOCs and Their Relevance to Software Engineering. Recuperado de: https://www.dre.vanderbilt.edu/~schmidt/PDF/ICSE-2014-MOOCS-Final.pdf

Tillmann, N. , Halleux, J., Xie, T. , Gulwani, S. , Bishop, J. (2013). Teaching and Learning Programming and Software Engineering via Interactive Gaming. 36th International Conference on Software Engineering, San Francisco, Estados Unidos. Recuperado de: http://research.microsoft.com/pubs/189240/icse13see-pex4fun.pdf

Pressman, R. (2010). Ingeniería del software, un enfoque práctico (Séptima edición). México: McGraw-Hill.

Vázquez Cano, E., López Meneses, E., Sarasola Sánchez-Serrano, J. (2013). La expansión del conocimiento en abierto: Los MOOC. España: Octaeadro.

Coursera (2011). Recuperado de: https://es.coursera.org/browse/computer-science?languages=es

Udacity (2011). Recuperado de: https://www.udacity.com/courses/software-engineering

EdX (2012). Recuperado de:
https://www.edx.org/course?search_query=software+engineering

Miríada X (2013). Recuperado de: https://miriadax.net/cursos

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá