

Introducción a MADS

En este apartado veremos una introducción a los contenidos de la asignatura y explicaremos su planificación. Tras leerlo tendrás una visión general de los temas de teoría y prácticas que vamos a abordar durante el cuatrimestre y conocerás el sistema de evaluación.

Datos de la asignatura

La asignatura **Metodologías Ágiles de Desarrollo de Software** es una asignatura optativa de 4º curso del Grado en Ingeniería Informática. Es una asignatura de la especialidad de **Ingeniería del Software**.

- **Departamento:** Ciencia de la Computación e Inteligencia Artificial.
- **Créditos ECTS:** 6 créditos ECTS, que se corresponden con 15 semanas de trabajo y 10 horas por semana, incluyendo clases y trabajo en casa (150 horas de dedicación en total).
- [Ficha UA de la asignatura](#).

El profesor de la asignatura es [Domingo Gallardo](#). Puedes consultarle enviándole una tutoría por UACloud o un correo electrónico a domingo.gallardo@ua.es. También puedes encontrarlo en Twitter con el usuario [@domingogallardo](#) y en [GitHub](#).

Materiales docentes

Todos los materiales de teoría de la asignatura están escritos en Markdown y publicados en el repositorio de GitHub [domingogallardo/mads-ua](#).

Los apuntes de las prácticas también están realizados en Markdown, subidos al repositorio [domingogallardo/practicas-mads](#), compilados con la herramienta [Material for MkDocs](#) y desplegados en GitHub Pages en [este enlace](#). El resultado es un sitio web donde es posible realizar búsquedas y navegar por secciones.

Utilizaremos Moodle como plataforma docente. Puedes encontrar la página de la asignatura en [este enlace](#) semana a semana los enlaces a los materiales en GitHub. También se realizarán en la plataforma las entregas finales de las prácticas (que se deberán ir subiendo también a GitHub). Y utilizaremos el foro para poner en común dudas que os puedan surgir cuando estáis realizando los trabajos.

Evolución de los contenidos de la asignatura

La asignatura está en constante evolución. Todos los años se revisan los contenidos, atendiendo a vuestro *feedback* y a los cambios en las tecnologías y en los conceptos impartidos.

Por ejemplo, podemos destacar los siguientes cambios que hemos introducido en los últimos años:

- Uso del *framework* Java Spring Boot, mucho más usado en la industria que el que utilizábamos anteriormente (*Play Framework*).
- Mayor énfasis en las funcionalidades de GitHub y de Git.
- Mayor importancia a las metodología XP y Kanban y menos tiempo dedicado a Scrum, debido a que ya lo habéis visto en otras asignaturas de la titulación.

Mejora continua y *Kaizen*

Queremos aplicar en el propio contenido de la asignatura una de las técnicas de las metodologías ágiles y metodologías *lean*: la **mejora continua** o **Kaizen**.



"Perfection is a direction, not a place"

Esta frase de Henrik Kniberg refleja muy bien la filosofía Kaizen de mejora continua que se aplica en las técnicas *lean* de gestión de empresas o en el desarrollo de software basado en entrega continua.

Queremos que la asignatura vaya evolucionando y mejorando curso a curso y que no se estanque en un sitio, por muy *perfecto* que nos pueda parecer.

Filosofía Kaizen

改善
kai = cambio zen = beneficio

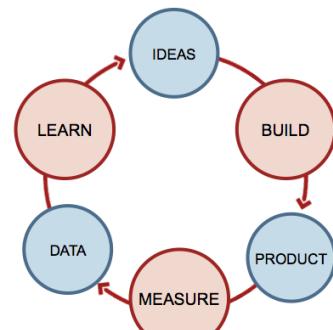
Kaizen (consultar [Kaizen](#) en la Wikipedia) es una palabra japonesa que significa mejora continua y cambio incremental. Un "Kaizen" es una pequeña mejora, que introduce un riesgo pequeño y que puede ser fácilmente implementada.

Las ideas Kaizen aplicadas a la mejora de procesos en la empresa tienen su origen en Japón en los años 50. En esos años se desarrolla en la empresa de fabricación de automóviles Toyota una metodología de gestión y fabricación totalmente distinta a la existente en Estados Unidos. Se trata del denominado **TPS (Toyota Production System)**. Su aplicación en las empresas japonesas puso a este país en la cabeza de la producción y venta de automóviles en los años 60 y 70 y constituyó todo un cambio de paradigma en la gestión. Esta nueva metodología fue analizada profusamente en las más importantes escuelas de negocios a lo largo de muchas décadas, dando lugar a lo que hoy se conoce como **métodos lean**.

En esta filosofía de mejora continua introducida por Toyota también se utiliza el denominado **ciclo PDSA (Plan-Do-Study-Act)** introducido por el estadístico Edwards Deming. Este ciclo trata de dirigir los cambios utilizando una metodología científica, en la que se realiza un plan definiendo alguna hipótesis, se realiza un experimento, se estudian los resultados obtenidos y se actúa aceptando o rechazando el cambio.

Este ciclo es muy utilizado en *startups* para construir y modificar el producto o software que se entrega al público.

Un ejemplo práctico de uso de la filosofía Kaizen se muestra en este [hilo de Twitter](#) de Dani Sánchez-Crespo [@DaniNovarama](#).



Henrik Kniberg

[Henrik Kniberg](#) es un divulgador muy importante de las metodologías ágiles. Ha trabajado con grandes empresas como Spotify y Lego para implantar en ellas estas metodologías. Le gusta compartir todo lo que

hace y muchos de los libros que ha escrito están en abierto. Vamos a utilizar muchos de sus materiales (diapositivas, vídeos, libros, artículos) en la asignatura.

Muy recomendable su [blog](#). Su Twitter es [@henrikniberg](#). Actualmente se ha apartado un poco del mundillo ágil y es desarrollador en Mojang, trabajando en las nuevas versiones de Minecraft.



Planificación y evaluación

Planificación MADS 2023-24

Semana	Fecha	Clases de teoría	Clases de prácticas	Entregas de práctica
Semana 1	13/9/2023	Sesión 1: Introducción a la asignatura	Práctica 1: Primera aplicación con Spring Boot	
Semana 2	20/9/2023	Sesión 2: Desarrollo de software	Práctica 2: Aplicación ToDoList	Entrega práctica 1 (19/9)
Semana 3	27/9/2023	Sesión 3: El estado Agile en 2020 Sesión 4: Manifiesto ágil		
Semana 4	4/10/2023	Sesión 5: Extreme Programming Sesión 6: Historias de usuario		
Semana 5	11/10/2023	Sesión 7: TDD		
Semana 6	18/10/2023	Sesión 8: Refactoring	Práctica 3: Integración con GitHub Actions y TDD	Entrega práctica 2 (17/10)
Semana 7	25/10/2023	Sesión 9: Diseño SOLID	Control de prácticas	
	1/11/2023	Fiesta		
Semana 8	8/11/2023	Control 1 teoría - 1, 2, 3, 4, 5, 6, 7, 8	Práctica 4: Trabajo en equipo con GitFlow y diseño de nuevas funcionalidades	Entrega práctica 3 (7/11)
Semana 9	15/11/2023	Sesión 10: Flujos de trabajo Git		
Semana 10	22/11/2023	Sesión 11: Desarrollo software Lean	Práctica 5: Sprint	Entrega práctica 4 (21/11)
Semana 11	29/11/2023	Sesión 12: Metodología Kanban		
	6/12/2023	Fiesta		
Semana 12	13/12/2023	Sesión 13: Continuous Delivery		
Semana 13	20/12/2023	Presentaciones de trabajos	Presentaciones de trabajos	Entrega práctica 5 (19/12)
	25/1/2024	Control 2 teoría - 9, 10, 11, 12, 13, 14		

En la figura anterior se puede ver un resumen con el cronograma del desarrollo de la asignatura.

La evaluación de la asignatura se divide en un 40% correspondiente a teoría y un 60% a prácticas.

Teoría (40%)

Los contenidos de la asignatura se dividen en 4 grandes bloques (el número de sesión corresponde al orden en el que se van a ver en la asignatura):

1. Valores y principios ágiles para el desarrollo del software (3 semanas, sesiones 1, 2, 3 y 4)
2. XP (eXtreme Programming) (4 semanas, sesiones 5, 6, 7, 8 y 9)
3. Metodologías lean y Kanban (2 semanas, sesiones 11 y 12)
4. Integración y entrega continua (2 semanas, sesiones 10 y 13)

La evaluación de los contenidos de teoría se hará de forma continua, mediante **2 exámenes de tipo test** que ponderarán un 15% cada uno y un **vídeo de divulgación** sobre algún tema de la asignatura que ponderará otro 10%.

El vídeo de divulgación deberás subirlo a YouTube y deberá tener una duración de entre 5 y 7 minutos. Deberás escoger un tema relacionado con la asignatura y presentarlo en un vídeo con un formato atractivo. Puede contener explicaciones con diapositivas, demostraciones de código, animaciones, etc.. Puedes aparecer o no en el vídeo, pero obligatoriamente debe estar **narrado por ti**.

Además del enlace a YouTube donde esté el vídeo publicado deberás entregar:

- Diapositivas, imágenes, código, etc. que aparezca en el vídeo.
- Guión o transcripción de la narración.
- Bibliografía y referencias que has usado.

Compartiremos una página en la que iréis apuntando los títulos de los vídeos y al final de la asignatura compartiremos todos los enlaces a YouTube.

Prácticas (40%)

Las prácticas ponderarán un 40% en la nota final de la asignatura.

Se realizarán cinco prácticas a lo largo de la asignatura. Las tres primeras serán **individuales** y las otras dos serán en **equipo de 3 personas**.

Las 5 prácticas y su ponderación en la nota final de prácticas (4 puntos) son las siguientes:

1. (10%, 0,4 puntos en la nota final de la asignatura) (Individual) Introducción a Spring Boot.
2. (25%, 1 punto) (Individual) Versión inicial de la aplicación ToDoList.
3. (25%, 1 punto) (Individual) Integración continua y TDD en Spring Boot.
4. (15%, 0,6 puntos) (Equipo) Integración continua con GitFlow y despliegue.
5. (25%, 1 punto) (Equipo) Desarrollo de una iteración usando integración continua, Kanban y prácticas ágiles.

Examen de prácticas (20%)

También se hará un examen práctico individual sobre las tecnologías y metodologías utilizadas en las prácticas. Tendrá una ponderación de un 20% en la nota final de la asignatura.

Exámenes

Las fechas de los exámenes y su ponderación en la nota final de la asignatura serán las siguientes:

- Examen 1 de teoría (15%, 1,5 puntos en la nota final de la asignatura): miércoles 8 de noviembre a las 11:00 sobre las sesiones 1 a 8.
- Examen 2 de teoría (15%, 1,5 puntos): En enero, en la fecha y hora en la que se realice el examen final de la asignatura, sobre las sesiones 9 a 13.
- Examen de prácticas (20%, 2 puntos): miércoles 25 de octubre, en las horas de prácticas.

Los exámenes de teoría serán de preguntas de tipo test. En el examen de prácticas se deberá trabajar con un repositorio GitHub con una aplicación Spring Boot y realizar una serie de modificaciones sobre ella utilizando la metodología de trabajo vista en clase de prácticas (commits frecuentes, issues, pull requests, versiones, etc.).

Bloque 1: Valores y principios ágiles

En la actualidad la palabra "Agile" está muy extendida en el mundo de los negocios, de las *startups* y de las empresas de software que quieren dar apariencia de que hacen las cosas bien. Muchos directivos y *managers* intermedios de grandes empresas hablan de "transformación digital" y de "agile". Abundan los congresos, conferencias y cursos sobre estos temas.

Sin embargo, cada vez hay más descontento precisamente entre los que comenzaron esta "revolución agile", entre los equipos de desarrollo. Tras el interés actual por "agile", ¿hay realmente un interés en hacer mejor las cosas? ¿Hay interés en no agotar a los equipos con jornadas interminables de trabajo? ¿En transferir parte del poder de decisión a los desarrolladores? ¿En realmente incorporar a los clientes en el ciclo de desarrollo? ¿En que el software desarrollado sea probado continuamente y que se entregue en incrementos pequeños que añaden nuevo valor? ¿En corregir y reordenar las prioridades conforme se aprende más del dominio?



Para apreciar realmente el significado de las metodologías ágiles hay que volver al principio, conocer los fundamentos y valores de estas metodologías. Si nos limitamos a seguir ciegamente unos procedimientos estándar (el *daily scrum*, el *sprint planning*, etc.) sin entender los valores y principios subyacentes nos convertiremos en *scrum zombies*.

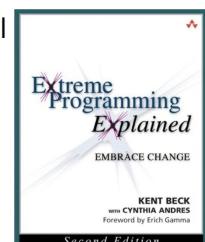
El bloque tendrá una **duración de 3 semanas** y veremos en él temas relacionados con los valores y principios ágiles

- Desarrollo del software, características y modelos, aceptación del cambio.
- Manifiesto ágil, repercusión del manifiesto en las metodologías de desarrollo, orígenes y fundamentos de las metodologías ágiles.

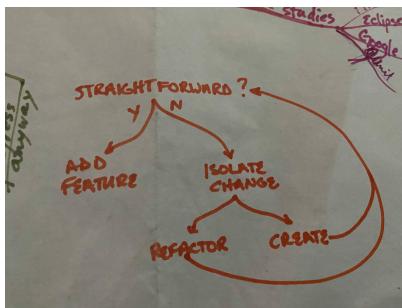
Bloque 2: XP (eXtreme Programming)

Extreme Programming (XP) fue una de las primeras metodologías ágiles. Kent Beck formula el conjunto de valores, principios y prácticas que constituyen esta metodología a finales de los 90 y principios del 2000.

A diferencia de otras metodologías, como Scrum o Kanban, la metodología está muy centrada en el desarrollo de software y muchas de sus prácticas, como *pair programming* o *TDD (Test Driven Design)*, son específicas para desarrolladores.



En este bloque incluiremos también otras técnicas y prácticas de desarrollo que, no siendo estrictamente parte de XP, son imprescindibles para poder gestionar el cambio. Un software bien diseñado y desarrollado debe ser fácilmente modificable y no debe acumular *parches* y soluciones parciales que lo van haciendo cada vez más incomprensible (la denominada *deuda técnica*).



© Kent Beck

Los tests y la refactorización son ideas fundamentales para tratar con el cambio sin ahogarse en la deuda técnica. Veremos técnicas para diseñar el software de forma que se facilite el cambio y para aislar las funcionalidades existentes de forma que las nuevas funcionalidades no entren en conflicto con las ya existentes.

El bloque tendrá una **duración de 4 semanas** y veremos en él los aspectos fundamentales de XP. Veremos en profundidad TDD, haciendo una demostración de este tipo de metodología de desarrollo. Hablaremos también de diseño SOLID y de refactorización.

- Valores y prácticas XP.
- Historias de usuario.
- TDD, Desarrollo Dirigido por los Tests.
- Diseño para el cambio: refactorización y SOLID.

Kent Beck

Kent Beck es un ingeniero de software, escritor, creador y divulgador de metodologías de diseño y desarrollo de software. Ha sido uno de los principales responsables del auge de las metodologías ágiles con el desarrollo y la divulgación de XP (EXtreme Programming).



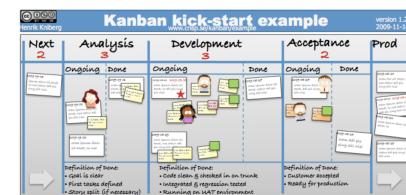
Además ha escrito sobre patrones de diseño de software, sobre testing, ha creado prácticas como TDD (*Test Driven Design*) o la más reciente TCR (*Test && Commit || Revert*) y ha desarrollado la librería de testing en Java jUnit.

Muchas ideas que veremos en la asignatura han sido propuestas o popularizadas por él.

- [Bibliografía de Kent Beck](#)
- [Charlas en YouTube](#)

Bloque 3: Metodologías lean y Kanban

Las metodologías *lean* son también una de las bases más importantes del cambio hacia una nueva filosofía y forma de trabajar. Son también una oportunidad de implicar a *managers* y directivos en la transformación ágil. Muchas empresas están aplicando con éxito estas técnicas para conseguir iteraciones rápidas y productos más útiles para sus clientes.



Veremos una introducción a estas metodologías, cómo aplicarlas al desarrollo de software y nos centraremos en la metodología Kanban.

Los objetivos principales de Kanban son la visualización y optimización del flujo de trabajo y en la detección de cuellos de botella en las distintas fases del desarrollo. Un elemento fundamental de la metodología es el uso de tableros en los que visualizar el estado de todas las tareas que está realizando el equipo.

El bloque tendrá una **duración de 2 semanas** y veremos los siguientes apartados:

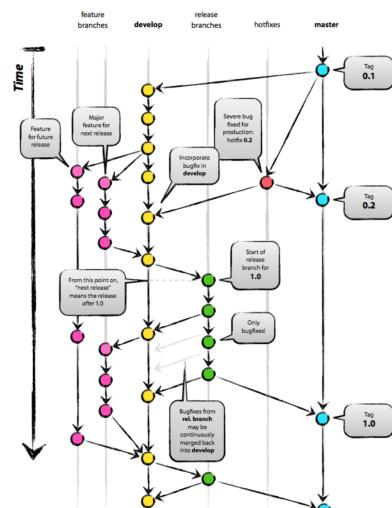
- Fabricación lean, empresa lean, desarrollo de software lean.
- Fundamentos de Kanban: visualización, ítems de trabajo, trabajo en progreso, flujo, mejora continua, límite en el WIP.
- Kanban vs. Scrum.

Bloque 3: Integración y entrega continua

Uno de los elementos importantes de las metodologías ágiles es la entrega continua al cliente de software que funciona. Para poder realizar esta entrega es fundamental poder añadir funcionalidades continuamente al software que estamos desarrollando y poder desplegarlo y ponerlo en manos del cliente en cuestión de minutos.

Los sistemas como Git y las herramientas de integración continua permiten trabajar con múltiples versiones del producto de forma segura y automatizar la ejecución de los tests y la construcción y despliegue de los ejecutables.

Este bloque tendrá una **duración de 3 semanas** y en él estudiaremos en profundidad Git y la plataforma GitHub con todas las funcionalidades que proporciona para realizar una integración continua de nuestro proyecto software: pull requests, revisión de código, scripts de compilación y despliegue. También estudiaremos principios generales que guían la entrega continua de software. Y herramientas como Docker que nos permitirá empaquetar nuestra aplicación en un contenedor que podrá ser desplegado y ejecutado en cualquier lugar.



- Git como sistema de control de versiones, flujos de trabajo con Git.
- Principios de la entrega continua de software.
- Herramientas y tecnologías para la integración continua, scripts de construcción y despliegue, infraestructura y entornos de integración, cloud.

Prácticas

Tecnologías de las prácticas

- [Spring Boot](#)
 - Framework de desarrollo rápido de aplicaciones web, usado en producción en múltiples empresas. Uno de los frameworks Java más demandados por el mercado.
 - Arquitectura MVC con controladores que responden a peticiones HTTP y generan vistas basadas en plantillas
 - Modelo de datos y acceso a BD con JPA
 - Integra la gestión de dependencias Maven
- [GitHub](#)
- [Git](#)

- [Docker](#)

Prácticas que desarrollaremos

- Práctica 1 (1 semana): **Primera aplicación** de introducción a Spring Boot.
- Práctica 2 (4 semanas): Aplicación **TodoList** en Spring Boot. Spring Boot, Git, GitHub. Metodología de trabajo Git y GitHub.
- Práctica 3 (3 semanas): **Integración continua**. Tests en Spring Boot y TDD. Integración continua con GitHub y Docker.
- Práctica 4 (2 semanas): **Configuración del trabajo en equipo**. Flujo de trabajo Git y GitHub modificado para el trabajo en equipo. Uso de pull requests.
- Práctica 5 (4 semanas): **Iteración de desarrollo**. Seleccionaremos las historias de usuario y las desarrollaremos durante una iteración de desarrollo, practicando las metodologías vistas en la asignatura. Mejoraremos también el tablero de GitHub para acercarlo más a un tablero de Kanban.

El desarrollo de software y la IA

El trabajo del desarrollador/a está siendo uno de los más impactados por la aparición de las nuevas herramientas y asistentes de IA, como Copilot o ChatGPT. Su uso se ha popularizado entre los profesionales como una forma de mejorar la productividad y minimizar las tareas aburridas y repetitivas, llevando a un descenso pronunciado de las consultas a sitios como *Stackoverflow* o el mismo Google.

Los modelos más avanzados como GPT-4 son incluso capaces de interpretar descripciones en lenguaje natural y generar el código necesario en el lenguaje de programación requerido.

Es previsible que con la aparición de modelos más potentes el impacto sea mucho mayor que en la actualidad, cambiando radicalmente el trabajo del desarrollo de software. Esto ya ha pasado alguna vez en la historia de la programación, por ejemplo cuando en los años 50 la introducción de lenguajes como Fortran, Lisp o Algol permitió expresar los programas usando un nivel de abstracción mucho mayor que el existente entonces del código ensamblador. Pero nunca hasta ahora ninguna tecnología había permitido generar, modificar y razonar sobre estos programas usando un agente que recibe instrucciones en lenguaje natural.

Estamos en un momento crucial de la historia de la programación y en una asignatura como la nuestra, en la que se habla de cómo mejorar el desarrollo de software, estamos obligados a reflexionar sobre ello. Por ello, en cada tema plantearemos un debate en el que discutiremos sobre el impacto de la IA en los aspectos concretos mencionados en el mismo.

Para comenzar, durante esta semana debéis leer el provocativo artículo de Henrik Kniberg: [Are Developers Needed in the Age of AI?](#). Lo debatiremos en clase la semana que viene.

Uso de herramientas y asistentes de IA en las prácticas de la asignatura

De la misma forma que es importante conocer todas las opciones y ayudas que nos proporcionan los IDEs modernos como IntelliJ o Studio Code, es también fundamental conocer las herramientas y asistentes de IA.

Os animo a que uséis estas herramientas como Copilot o ChatGPT en el desarrollo de las prácticas. La mejor forma de conocer su utilidad y sus limitaciones es usándolas. Y también pueden ser de mucha utilidad para aprender y resolver dudas sobre los conceptos, lenguajes y frameworks usados en la asignatura.

Eso sí, no debéis olvidar que el objetivo final de la asignatura es aprender estos conceptos y que deberéis demostrarlo en unos exámenes en los que no se permitirá el uso de estas herramientas.

Bibliografía

A continuación puedes ver un listado de libros en los que ampliar los contenidos de la asignatura. Se incluye un enlace a O'Reilly para poder consultarlos con tu ordenador o tablet. Puedes encontrar todos los libros en la Playlist de O'Reilly [Metodologías Ágiles](#).

La mayoría también está disponible en formato físico en la biblioteca de la UA. Puedes consultar su ubicación y disponibilidad en el [catálogo de la UA](#)

Para acceder a la cuenta de O'Reilly debes darte de alta con el correo electrónico de la UA. Puedes encontrar más información en [este enlace](#).

Agile en general y Scrum

- Green Stellman: [Learning Agile](#)
- James Shore: [The Art of Agile Development](#)
- Jonathan Rasmusson: [The Agile Samurai](#)
- Mike Cohn: [Succeeding with Agile](#)
- Mike Cohn: [Agile Estimating and Planning](#)
- Mike Cohn: [User Stories Applied](#)
- Kenneth S. Rubin: [Essential Scrum](#)
- John Reffries: [The Nature of Software Development](#)

EXtreme Programming

- Kent Beck: [Extreme Programming Explained \(Primera edición\)](#)
- Kent Beck: [Extreme Programming Explained \(Segunda Edición\)](#)
- Kent Beck, [Test Driven Development By Example](#)
- Henrik Kniberg: [Scrum and XP from the Trenches \(2nd edition\)](#)

Buenas prácticas de desarrollo

- David Thomas; Andrew Hunt: [The Pragmatic Programmer: your journey to mastery, 20th Anniversary Edition, 2nd Edition](#)
- Scott Millett: [Patterns, principles, and practices of domain-driven design](#)
- Michael Feathers: [Working Effectively with Legacy Code](#)
- Martin Fowler: [Refactoring](#)
- Robert C. Martin: [Clean Code](#)
- Robert C. Martin: [Clean Architecture: A Craftsman's Guide to Software Structure and Design](#)

Lean y Kanban

- Mary Poppendieck and Tom Poppendieck: [Implementing lean Software Development: From Concept to Cash](#)
- Henrik Kniberg: [Lean from the Trenches](#)
- Hammarberg, Sundén: [Kanban in Action](#)

Continous delivery, integration and deployment

- David Farley and Jez Humble: *Continuous delivery*
- Jez Humble, Gene Kim and Nicole Forsgren: *Accelerate*
- Scott Chacon: *Pro Git*
- Brent Laster: *Professional Git*
- Jeffrey Nickoloff, Stephen Kuenzli: *Docker in Action*