



# Procesos de Desarrollo de Software

Prof. Juan Pablo Sandoval  
[juanpablo.sandoval@ing.puc.cl](mailto:juanpablo.sandoval@ing.puc.cl)

A	B	C	D	E	F	G	H
		Clase Martes		Clase Jueves		Ayudantía Viernes	Proyecto
Semana	Fecha	Tema	Fecha	Tema	Fecha	Tema	
1	7/3/22	Introducción a la Ingeniería de Software	9/3/22	Presentación y Ruby	11/3/22	-	Formulario inscripción Grupos (7/3/22)
2	14/3/22	Procesos de Desarrollo de Software	16/3/22	Ruby y Web	18/3/22	Ruby	Publicación Grupos (18/3/22)
3	21/3/22	Scrum	23/3/22	Ruby On Rails y MVC	25/3/22	Troubleshooting	
4	28/3/22	Requisitos, relatos de usuario	30/3/22	CRUD en Ruby On Rails	1/4/22	Routing y CRUD	
5	4/4/22	Planeación y Estimaciones	6/4/22	Tests Unitarios (RSpec)	8/4/22	Testing	Sprint 0 (8/4/22)
6	11/4/22	Estimaciones	13/4/22	Active Record	15/4/22	-	
7	18/4/22	Kanban, Casos de Uso	20/4/22	Asociaciones	22/4/22	Devise	
	22 de Abril Interrogación 1 18:30 horas						
8	25/4/22	Diseño, UML	27/4/22	UML	29/4/22	Sala de Ayuda	Sprint 1 (29/4/22)
9	2/5/22	No hay clases (Planif Escuela)	4/5/22	Equipos de Trabajo	6/5/22	A + M	
10	9/5/22	Patrones de Diseño I	11/5/22	Práctica UML	13/5/22	Patrones	
11	16/5/22	Patrones de Diseño II	18/5/22	Práctica Patrones de Diseño	20/5/22	Sala de Ayuda	Sprint 2 (20/5/22)
12	23/5/22	Documentación	25/5/22	Práctica Patrones de Diseño	27/5/22	-	
	27 de Mayo Interrogación 2 18:30 horas						
13	30/5/22	Arquitectura	1/6/22		3/6/22	Arquitectura	
14	6/6/22	Patrones y Arquitecturas comunes	8/6/22	Documentación	10/6/22	Sala de Ayuda	Sprint 3 (10/6/22)
15	13/6/22	SQA 1	15/6/22	SQA 2	17/6/22	-	
16	20/6/22	Feriado UC	22/6/22		24/6/22	Presentaciones	Presentaciones
17	27/6/22	Feriado Nacional	29/6/22	Semana Exámenes		Semana Exámenes	Semana Exámenes
		6 de Julio Examen Final 8:30 horas		Semana Exámenes		Semana Exámenes	Semana Exámenes

---

# Repaso Clase 2



## Visto en la clase anterior

- Como crear objetos en Ruby
  - clases, métodos, constructores, loops
  - visibility/scope (private, protected, public)
- Tips para analizar/mejorar la calidad de su código
  - mínimo acoplamiento / máxima cohesión
  - evitar los diferentes code-clones
    - se puede eliminar el código duplicado encapsulando lo que varía de la parte similar ya sea en un método, o clase.



## Repaso rápido

- Porque se recomienda inicialmente tener los atributos/métodos como privados?
- Porque es recomendable depender menos de otras clases/módulos?
- Cual es el problema con tener código duplicado?



## Preguntas sobre Ruby

- Soporta herencia múltiple? No, y aunque así lo hiciera no es recomendable usar herencia múltiple.
- Cómo heredar una clase de otra?

```
class B < A
  ...
  ...
end
```

---

# Repaso Clase 1



# Principales desafíos del desarrollo del software






# Principales desafíos del desarrollo del software

- cumplir los plazos
- calidad que no hayan bugs
- estimar costo de mantención
- el software se tiene que adaptar
- seguro
- portabilidad
- fácil de utilizar
- buena usabilidad escalabilidad



## Cual es el problema?

- software que a menudo no hace lo que usuarios quieren
- software que es muy caro
- software que no es suficientemente rápido
- software que es difícil de usar
- software que no puede ser portado
- software que es muy caro de mantener
- software que es poco confiable
- proyectos de software que se atrasan



One of the planning documents for software research revealed --in a parenthetical remark only-- an unchallenged tacit assumption by referring to **"the tradeoff between cost and quality"**. Now in all sorts of mechanical engineering it may make sense to talk about "the tradeoff between cost and quality", **in software development this is absolute nonsense**, because poor quality is the major contributor to the soaring costs of software development.

**Dijkstra, EWD690**



# Procesos de Desarrollo de Software

Prof. Juan Pablo Sandoval  
[juanpablo.sandoval@ing.puc.cl](mailto:juanpablo.sandoval@ing.puc.cl)



## Componentes de un proceso

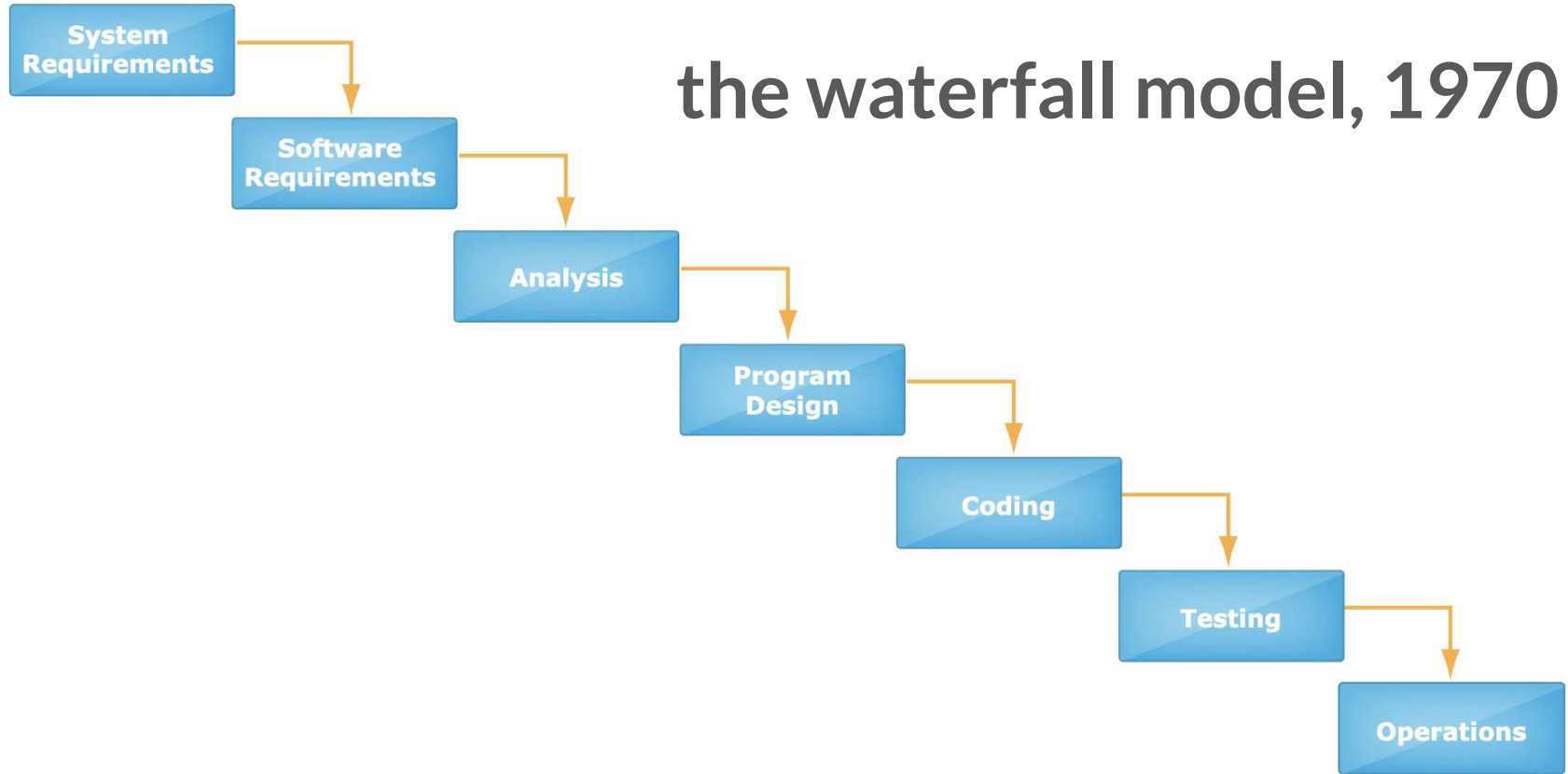
- actividades
- personas
- herramientas
- entradas
- salidas



## Proceso de Desarrollo

- La estrategia usada para llevar a cabo las actividades de desarrollo hasta llegar al producto
- Cada actividad tiene sus insumos y entregables, puede requerir herramientas
- Una actividad puede aparecer más de una vez en el proceso
- Existen diversos modelos de proceso que pueden ser usados

# the waterfall model, 1970



from: Winston Royce, "Managing the Development of Large Software Systems",  
Proceedings of IEEE WESCON 26 (August): 1–9, 1970.




## El modelo en cascada

- Cómo se caracteriza secuencia lineal de actividades una detrás de otra
- La principal ventaja es que es fácil de entender
- Desventajas:
  - se dilata demasiado la reducción del riesgo
  - muy difícil incorporar cambios en los requisitos
- Podría recomendarse en proyectos cortos

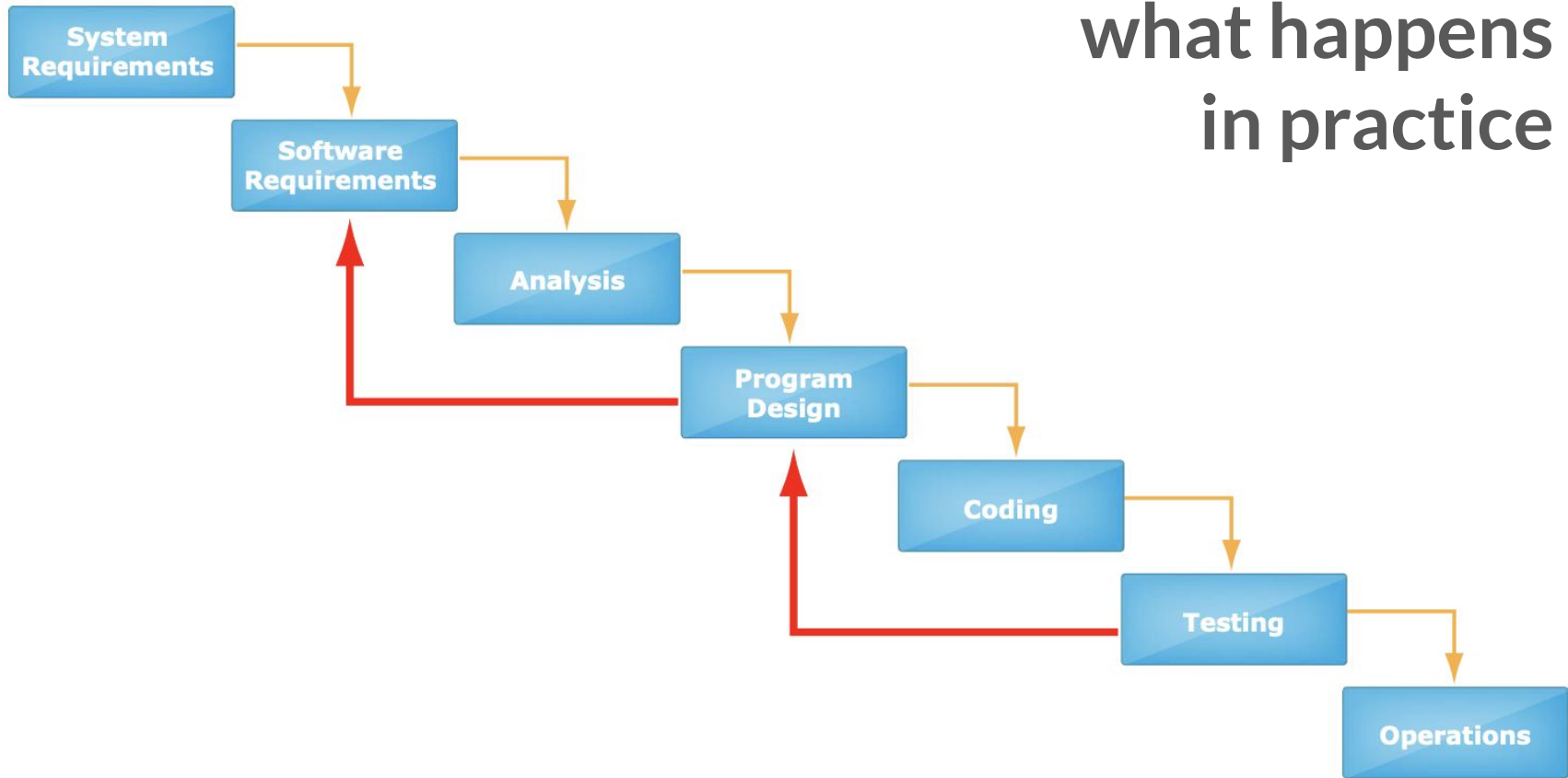


# what Royce actually said



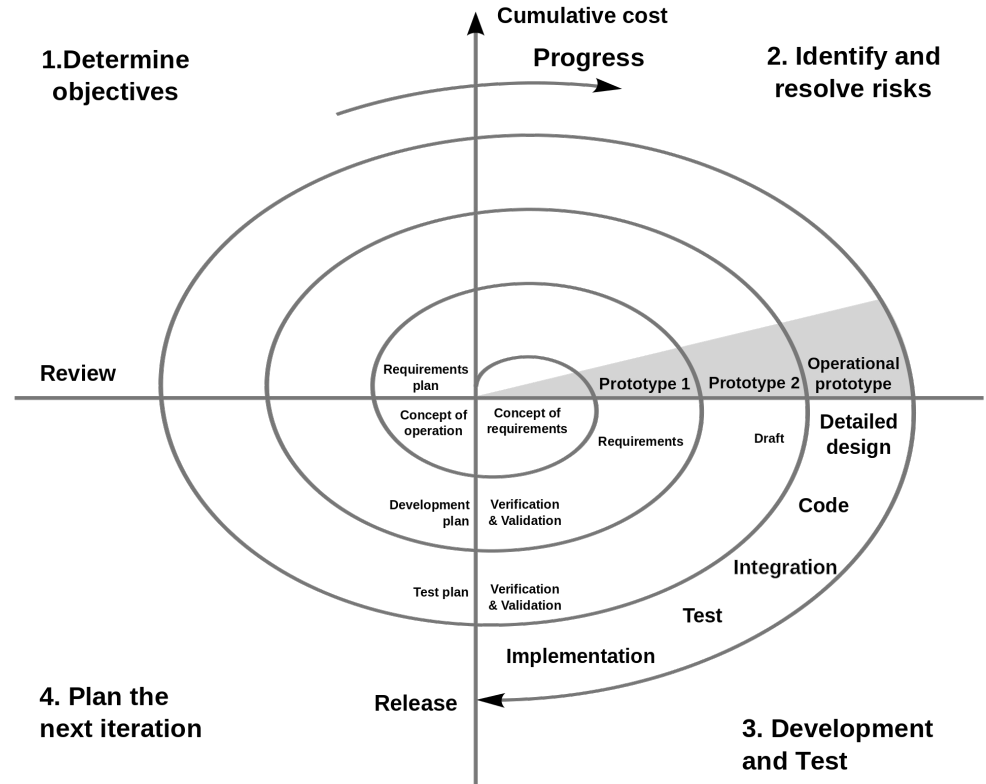
*The testing phase which occurs at the end of the development cycle is the first event for which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed. These phenomena are not precisely analyzable. They are not the solutions to the standard partial differential equations of mathematical physics for instance. Yet **if these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required.** A simple octal patch or redo of some isolated code will not fix these kinds of difficulties. **The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated.***

# what happens in practice



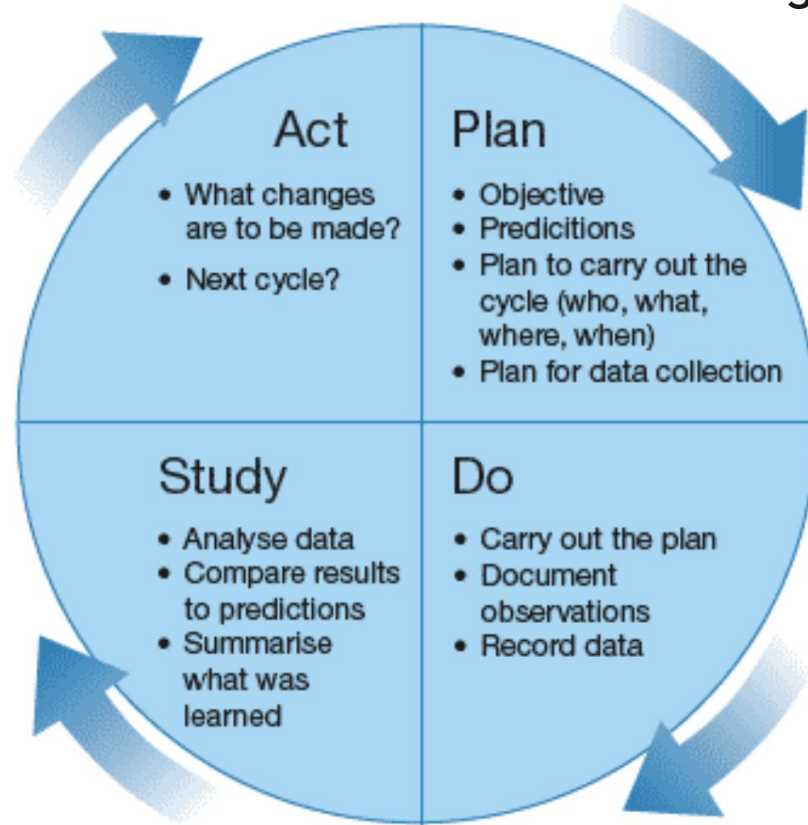
Waterfall model in practice, iterations are not restricted to successive steps.

# spiral model, 1986



Boehm, B (July 2000). "Spiral Development: Experience, Principles, and Refinements" (PDF). *Special Report*. Software Engineering Institute. CMU/SEI-2000-SR-008.

## origins of iterative approaches



### plan-do-study-act

- › Shewart, 1930s
- › Deming, 1940s

### Project Mercury

- › NASA, 1960s
- › half-day iterations
- › tests before each iter
- › became IBM Federal Systems Division

Craig Larman, Victor R. Basili (June 2003). "Iterative and Incremental Development: A Brief History". IEEE Computer(IEEE Computer Society) 36 (6): 47–56.

# UML AND THE UNIFIED PROCESS

PRACTICAL OBJECT-ORIENTED  
ANALYSIS & DESIGN

JIM ARLOW  
ILA NEUMAN

## THE UNIFIED SOFTWARE DEVELOPMENT PROCESS

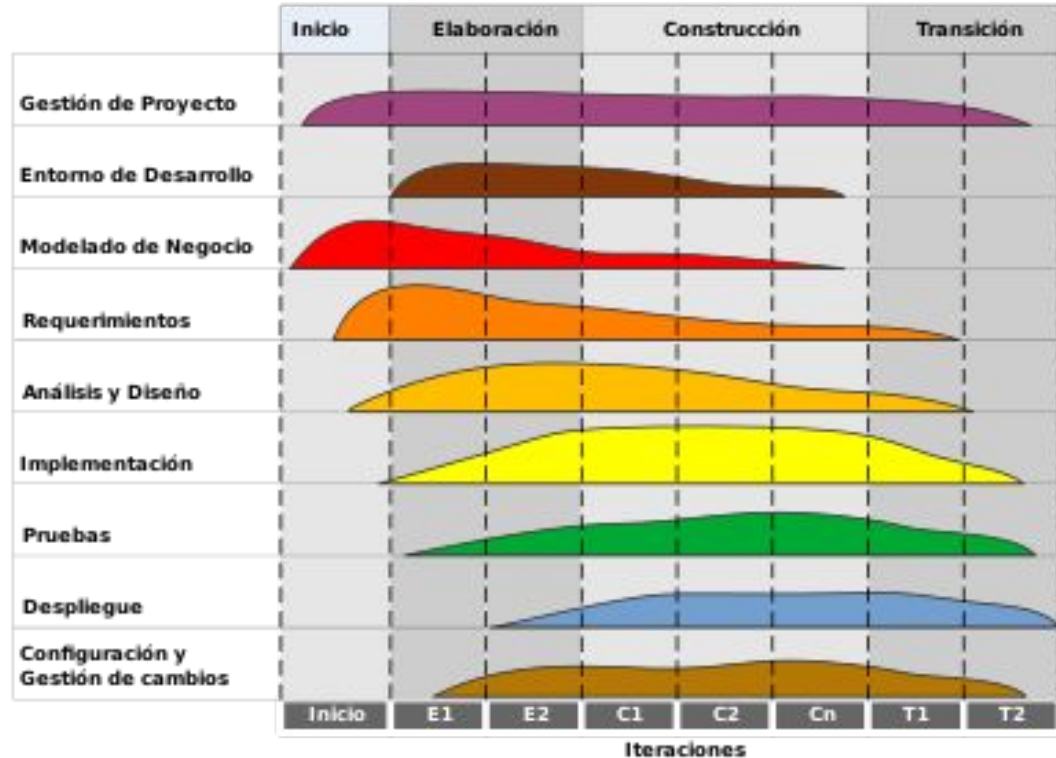
IVAR JACOBSON  
GRADY BOOCH  
JAMES RUMBAUGH

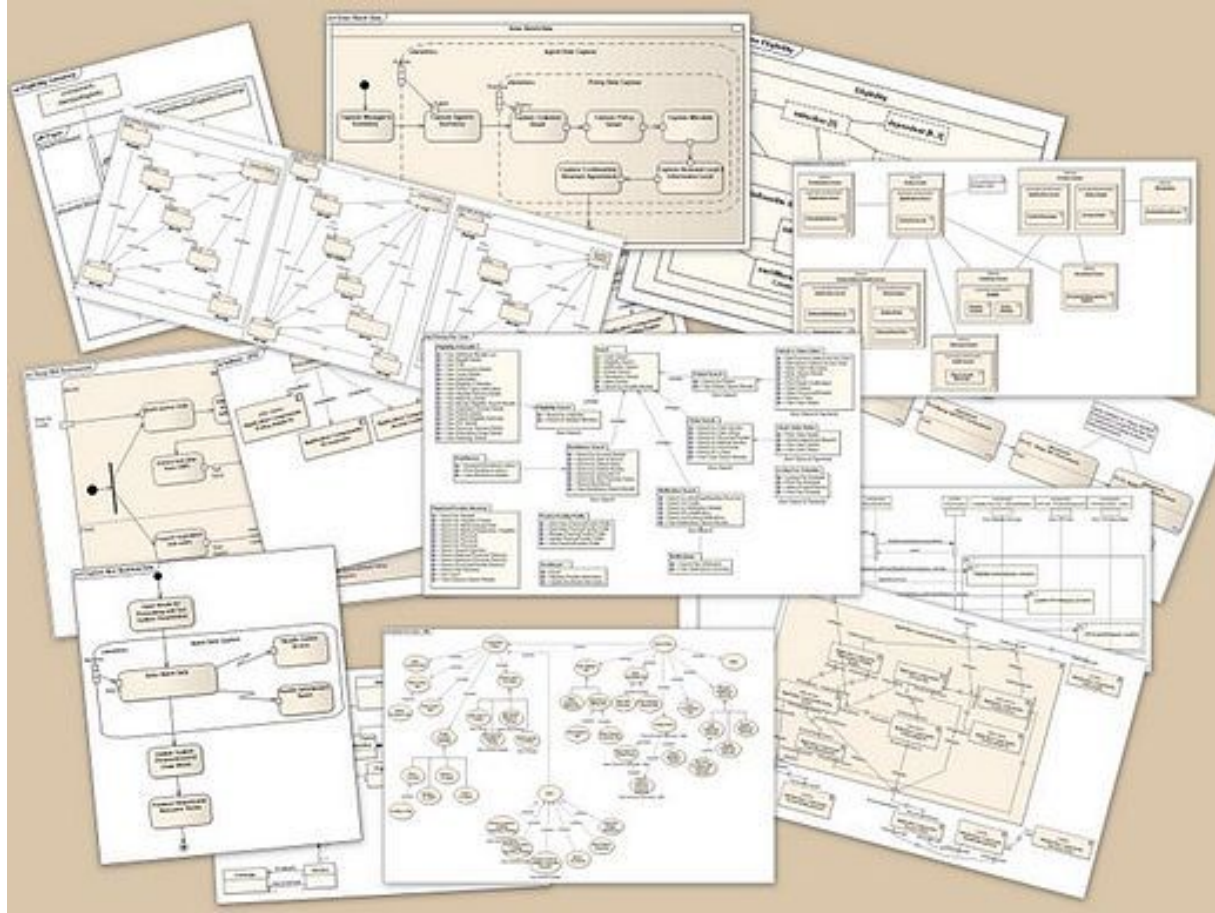


*The complete guide  
to the Unified  
Process from the  
original designers*



# Unified Process





# Unified Model Language



# extreme programming

- **Kent Beck, 1999**
  - take best practices to “extreme” levels
  - developed during C3 project with Ron Jeffries
- **a sample of XP practices**
  - test first: acceptance and unit tests
  - continuous integration
  - pair programming
  - repeated refactoring
- **Chrysler’s C3 payroll system**
  - started in 1996, cancelled in 2000
  - implemented in Smalltalk
  - running payroll took 1000 hours initially
  - Chrysler said they abandoned XP after this



# Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck  
Mike Beedle  
Arie van Bennekum  
Alistair Cockburn  
Ward Cunningham  
Martin Fowler

James Grenning  
Jim Highsmith  
Andrew Hunt  
Ron Jeffries  
Jon Kern  
Brian Marick

Robert C. Martin  
Steve Mellor  
Ken Schwaber  
Jeff Sutherland  
Dave Thomas



  
**Pero, qué es Agile?**






iterativo e incremental

**Dar más valor a ....**

<b>individuos y interacciones</b>	que a procesos y herramientas
<b>software funcional</b>	que la documentación comprensiva
<b>colaboración con el cliente</b>	que a la negociación del contrato
<b>respuesta al cambio</b>	que seguir el plan

- 
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

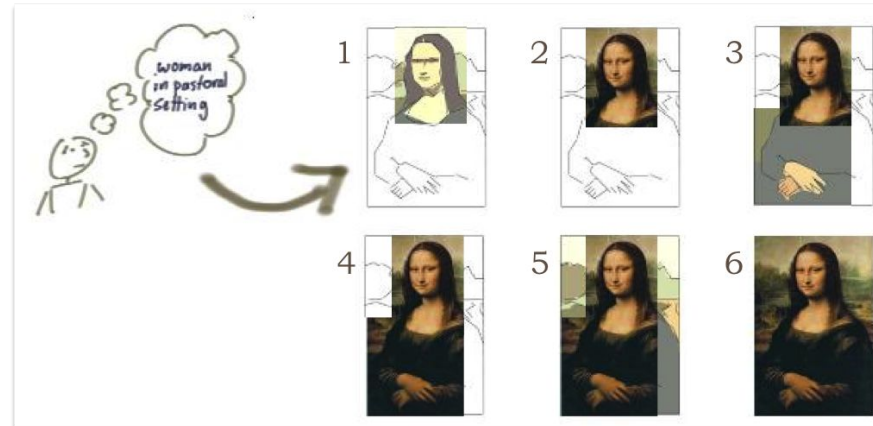


---

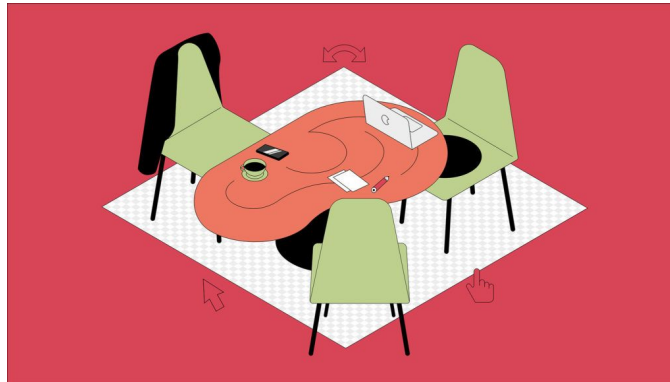
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.




3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.



4. Business people and developers must work together daily throughout the project.





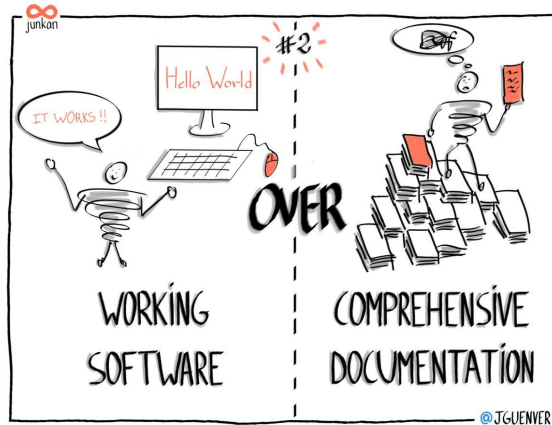
5. Build projects around motivated individuals.  
Give them the environment and support they  
need, and trust them to get the job done.


6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.






## 7. Working software is the primary measure of progress.






8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.




8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.




9. Continuous attention to technical excellence and good design enhances agility.



10. Simplicity – the art of maximizing the amount of work not done – is essential.



11. The best architectures, requirements, and designs emerge from self-organizing teams.



12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

---

**Agile is a set of values and principles**





**Escenario:** estás implementando una feature pequeña y te das cuenta que necesitas crear la base de datos.

Piensas que ya que vas a diseñar y crear la base de datos. Es mejor crear todas las tablas que necesitas de una sola vez.

**Estamos siendo ágiles?**

---

**Prácticas**

**Xtreme Programming - XP**



## Entre las prácticas más populares de XP están:

- Pair programming
- **Test Driven Development**
- Continuous Integration
- **Code Refactorings**
- Small releases
- On-site customer



**La siguiente clase veremos Scrum & Kanban**