Acerca de la Ingeniería de Software, el subdesarrollo, y otras cuestiones

IX Jornada de Gerencia de Proyectos
Marzo 24, 2011
Bogotá, Colombia

Jorge Aramburo Siegert
CEO, PSL

Precaución!

Esta presentación contiene opiniones

Propósito de la presentación

La presentación tiene cuatro propósitos fundamentales:

- **Reflexionar un poco sobre lo que está aconteciendo con la “Ingeniería de software” en el mundo.**
- **Explorar brevemente algunas prácticas de la ingeniería en Latinoamérica.**
- **Sembrar algunas inquietudes acerca del papel que debería jugar Latinoamérica en la evolución que se está dando en la profesión.**
- **No aburrirlos (la presentación es muy corta).**

Contexto

Los siguientes son algunos de los logros de PSL. *Solo se presentan para darle contexto a las reflexiones que planteamos.*

- **Primera y única latinoamericana en alcanzar el nivel 5 en CMM (2002)**
- **Octava en el mundo en lograr el nivel 5 de CMMI (2003).**
- **Primera compañía de TI en la región en certificarse en ISO 9001 hace mas de once años.**
- **Certificada en ISO 27001 (seguridad de la información).**
- **Ganadora del IEEE / SEI Software Process Achievement Award**
- **Ganadora el premio a la excelencia en software 2010 otorgado por el SEI.**
- **Adoptante temprano de CMMI for Services, ITIL y eSCM.**

La historia sin fin ...

1945 - 1965	Codifique y arregle. Codifique mas y arregle mas.
1965 - 1985	La “crisis del software”.
1969	En la Conferencia de la OTAN en Alemania se acoge el término “Ingeniería de Software”.
1970	“Managing the development of large software systems” por Winston Royce, dió origen al modelo en cascada.
1986	Los japoneses Hirotaka Takeuchi e Ikujiro Nonaka describen una nueva forma de organizar y gestionar grupos para desarrollo de nuevos productos (SCRUM).

La historia sin fin ...

1989	"Managing the software Process" por Watts Humphrey. Se convirtió en el fundamento de CMM y su posterior evolución, CMMI.
1990's	UP, RUP. Se comenzaron a gestar XP, SCRUM aplicado al desarrollo de software, Lean y otras.
2001	El manifiesto ágil.
2009 – 2010	SEMAT: Volvamos a pensar en esto.

¿Sabemos realmente que es Ingeniería de Software?

Los dos extremos

A raíz del indudable éxito y utilidad de las aproximaciones Ágiles, en los últimos años se ha planteado una discusión entre …

- Ceremoniosas
- Pesadas
- Prescriptivas
- Predictivas
- Con una cultura de comando y control
- Responsabilidad solo del grupo de ingeniería
- Guiadas por planeación tradicional

Poco abordan el componente humano

- Ágiles
- Livianas,
- Adaptativas
- Centradas en grupos multidisciplinarios y multifuncionales
- Responsabilidad colectiva de todos lo miembros

Con buena comprensión del papel del ser humano en el éxito de los proyectos

¿Y ... por qué la ceremonia?

Las aproximaciones tradicionales, ceremoniosas, pesadas, han encontrado fundamento, entre otras cosas, en...

- **El SEI (Software Engineering Institute), el modelo CMM y muchos Lead Assessor**
  - **CMM fue desarrollado para cumplir las exigencias del DoD de los EE.UU., en *proyectos a precio fijo*, de alta complejidad, altísimo riesgo, para desarrollar el componente de software de sistemas mayores (misiles, equipos médicos, aeronaves, etc.).**
  - **Los lead assessor fueron educados para proceder de acuerdo con el modelo, es decir, de acuerdo con las exigencias de DoD. Las prácticas sugeridas se convirtieron en obligatorias y las áreas de proceso en procesos.**
  - **El SEI permitió la expansión de CMM / CMMI incorrectamente aplicados.**

¿Y … por qué la ceremonia?

Las aproximaciones tradicionales, ceremoniosas, pesadas, han encontrado fundamento, entre otras cosas, en…

- **El SEI (Software Engineering Institute), el modelo CMM y muchos Lead Assessor**
  - **El SEI no le prestó verdadera atención a las aproximaciones ágiles hasta fines del 2008. Solo la versión 1.3, liberada hace unos días (Octubre 29 / 2010), incorpora esas aproximaciones en todo el modelo.**
- **El modelo en cascada (incorrecta interpretación del documento de Royce).**
- **Los proyectos a precio fijo, y las compras utilizando solo RFP y procesos ceremoniales impersonales.**
- **Mala práctica profesional de compradores y vendedores.**

¿Y … por qué la ceremonia?

Las aproximaciones tradicionales, ceremoniosas, pesadas, han encontrado fundamento, entre otras cosas, en…

- El SEI (Software Engineering Institute), el modelo CMM y muchos Lead Assessor
  - El SEI no le prestó verdadera atención a las … fines del 2008. Solo la … unos días (Octubre 29 / 2010), … nes en todo el modelo.
  - … cta interpretación del … documento de Roy…

El precio fijo obliga la utilización de modelos pesados, con mucha documentación y lentos. Cuando resultan bien, pocas veces, el cliente obtiene el sistema que contrató, pero no el que realmente necesita

- **Los proyectos a precio fijo, y las compras utilizando solo RFP y procesos ceremoniales impersonales.**
- **Mala práctica profesional de compradores y vendedores.**

¿Y… por qué la ceremonia?

El proyecto es estratégico para la compañía [XXXX]….

- **Las preguntas deben enviarse por escrito…**
- **No se aceptan reuniones con los proponentes….**
- **Por favor enviar las hojas de vida de quien desarrollará el proyecto en caso de ser seleccionado….**
- **Al momento no se tienen requerimientos en detalle, el proveedor debe levantarlos…**
- **Enviar cronograma detallado y el costo total….**
- **Si el proyecto toma menos tiempo se pagará menos, si toma mas se pagará lo ofertado…**
- **Anexar procesos de desarrollo, administración de la configuración, pruebas, entrenamiento, etc., etc.**

Y si… hay proponentes … y muchos problemas.

Las aproxi… han encont…

- El SEI (So… muchos L…
  - El SEI … aproxi… versió… incorp…
- El model… documen…
- **Los proyectos a… …jo, y las compras utilizando solo RFP y procesos cer… …nales impersonales.**
- **Mala práctica profesional de compradores y vendedores.**

¿Y ... por qué la ceremonia?

Las aproximaciones tradicionales, ceremoniosas, pesadas, han encontrado fundamento, entre otras cosas, en...

- **La utilización de CMM/CMMi, ISO, etc., como estrategia de mercadeo, no como un vehículo de mejoramiento**
- **La mala aplicación del cuerpo de conocimientos del PMI**
- **La incorrecta aplicación de ISO 9001, ITIL, etc.**
- **Consultoría inadecuada, corrupción, desconfianza**
- **Falta de reflexión. Las personas / organizaciones "compran" lo que mejor se mercadea**
- **Ingenuidad con buenas intenciones, sobre todo de organizaciones gremiales.**
- **Poca comprensión de las aproximaciones ágiles**

¿Y ... por qué la ceremonia?

Las aproximaciones tradicionales, ceremoniosas, pesadas, han enc

- La uti
  merca
- La ma
- La inc
- Consultoría inad[illegible]apción, desconfianza

- **Falta de reflexión. Las personas / organizaciones "compran" lo que mejor se mercadea**
- **Ingenuidad con buenas intenciones, sobre todo de organizaciones gremiales.**
- Poca comprensión de las aproximaciones ágiles

- **Tome el curso de [ITIL, CMMI, PMI, PSP, TSP, etc.] y le garantizamos la certificación!**
- **No se quede atrás de sus competidores, certifíquese en [YYY]**
- **¿Ha perdido información valiosa? El modelo / estándar [XXXX] tiene la solución perfecta para su compañía!**
- **[Nombre del gremio] está desarrollando la metodología para el [nombre del país]. Así lograremos ser mas competitivos!**
- **Asista a la IX Jornada y …. (estas conferencias son informativas, no formativas. Reflexione y estudie sobre lo que le interesa. Sirven también para hacer relaciones).**

¿Entonces...?

Trabaje con los modelos, no para ellos

El objetivo es el mejoramiento para crear el software que requiere el mundo y las organizaciones!

"Todavía creo que hace sentido hacer ingeniería de software. Pero eso no es exactamente lo que ingeniería de software ha llegado a significar. El término abarca un conjunto específico de disciplinas incluyendo procesos definidos, inspecciones y walkthroughs, ingeniería de requerimientos, matrices de trazabilidad, métricas, control de calidad preciso, planeación y control rigurosos y codificación y documentación estándar.

... El desarrollo de software es y siempre será de alguna manera experimental. La construcción en si del software no, pero su concepción si. Y aquí es donde nuestro foco debe estar".

Tom DeMarco

IEEE, Agosto 2009

¿Entonces...?

SEMAT (Software Engineering Method and Theory)

Software engineering is gravely hampered today by immature practices. Specific problems include:

- **The prevalence of fads more typical of fashion industry than of an engineering discipline.**
- **The lack of a sound, widely accepted theoretical basis.**
- **The huge number of methods and method variants, with differences little understood and artificially magnified.**
- **The lack of credible experimental evaluation and validation.**
- **The split between industry practice and academic research.**

¿Entonces...?

SEMAT (Software Engineering Method and Theory)

We support a process to refound software engineering based on a solid theory, proven principles and best practices that:

- **Include a kernel of widely-agreed elements, extensible for specific uses**
- **Addresses both technology and people issues**
- **Are supported by industry, academia, researchers and users**
- **Support extension in the face of changing requirements and technology.**

No sabemos que sucederá, pero la iniciativa es una muestra más de lo poco maduro que es el campo de la ingeniería de software

Hemos aprendido

Independientemente de la aproximación al desarrollo que utilice, sus valoraciones, certificaciones, experiencia y poder económico, la primera condición para el éxito reside en las personas, los propósitos, valores y cultura de su organización

"El paradigma las "Personas sobre los procesos" es muy importante, hasta que usted cae en cuenta que su grupo consiste de dos trolls, un loro, una peluquera y un relativamente brillante gerente de proyecto que ocurre que es ciego, sordo y mudo…. Ninguna cantidad de coaching puede hacer que ese grupo desarrolle un producto exitoso"

Jurgen Appelo
Management 3.0

Hemos aprendido

Todos sabemos que un mal cliente es tan perjudicial para un proyecto como un mal grupo de ingeniería. Infortunadamente a los clientes no se les puede pedir hoja de vida (en Suramérica), y raramente se les puede cuestionar porque "el cliente siempre tiene la razón (lo cual no es cierto)".

- **Las compañías y los grupos internos de TI, deben estar preparados para manejar clientes conflictivos, irrazonables y mal preparados.**
- **Es necesario aprender a decir "no", a tiempo. No haga proyectos que sabe van a ser un problema!.**
- **Asegúrese de llegar a acuerdos razonables antes de aceptar un proyecto.**
- **Si el contrato es muy complicado y detallado, tenga cuidado. Seguramente ese cliente ha tenido malas experiencias y se fue para el otro extremo.**

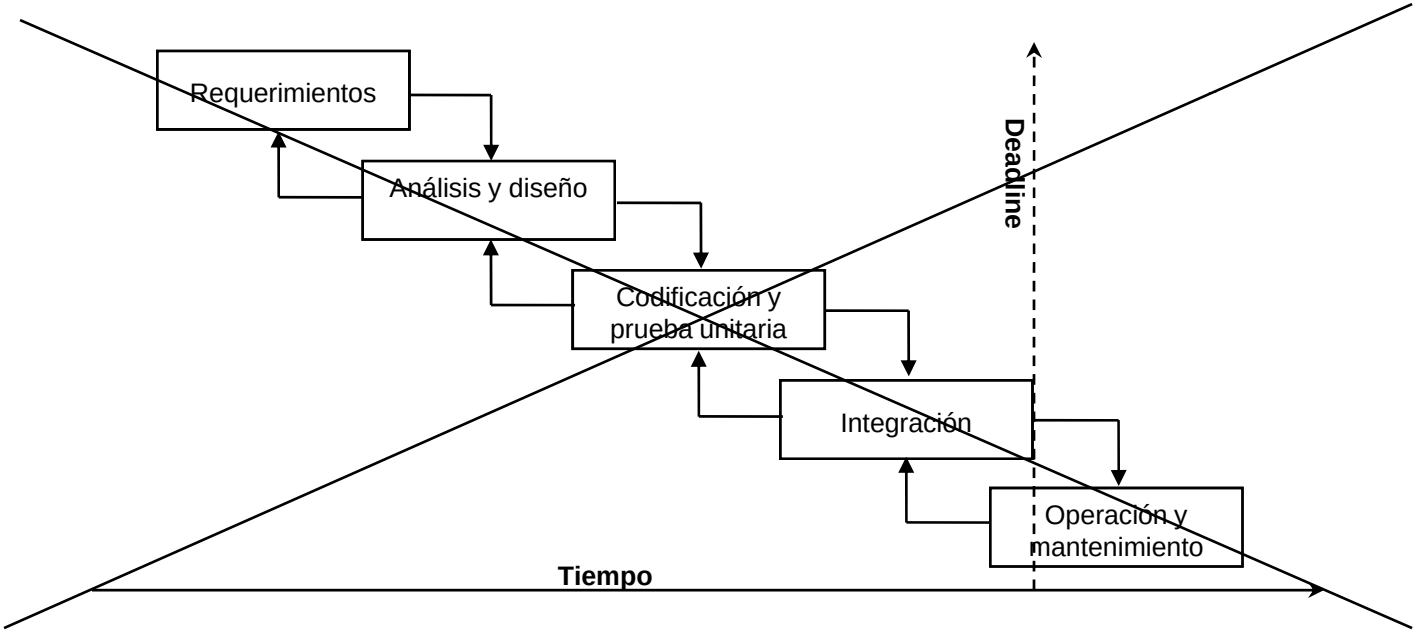
Hemos aprendido

CMMi es muy útil, si es correctamente implementado

- **Es metodológicamente agnóstico.**
- **Reconoce, por fin, las aproximaciones ágiles, y les da un gran despliegue en la última versión. La versión 1.3 tendrá un gran impacto pues obligará a cambiar de óptica a muchos lead assessors, e impulsará la adopción de aproximaciones ágiles en el mundo entero.**
- **Provee un contexto organizacional para el mejoramiento, lo que no ocurre con las aproximaciones al desarrollo, las cuales solo se centran, con excepción tal vez de Scrum, en el ciclo de vida del software.**
- **Provee muy buenas guías en áreas de proceso no abordadas por las aproximaciones metodológicas.**
- **Contiene mejores prácticas para proyectos riesgosos, grandes, no colocados.**

Hemos aprendido

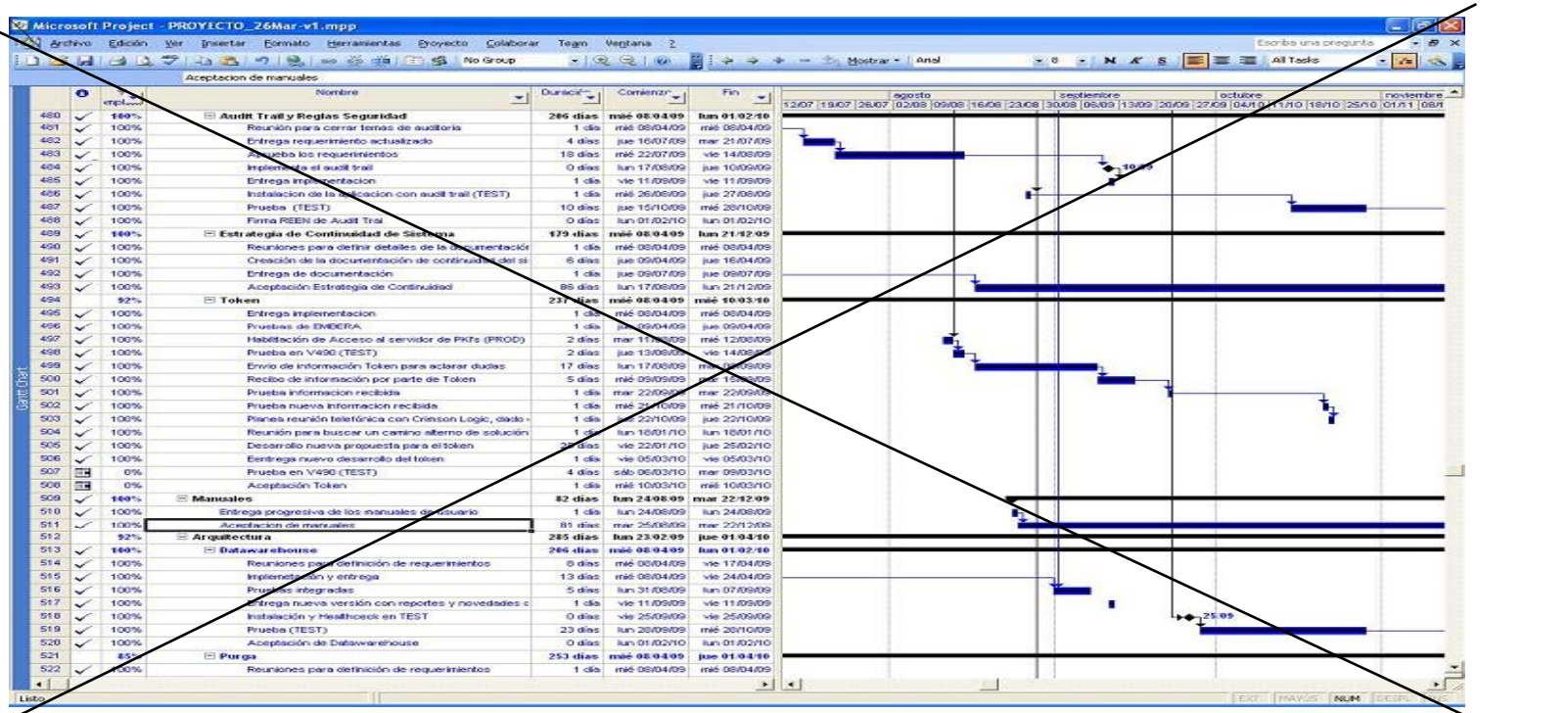
A juicio del autor, la aproximación en cascada no es apropiada para desarrollar software. Si tiene que utilizarla conozca bien sus riesgos para administrarlos!



El software debe desarrollarse de manera iterativa e incremental, en iteraciones cortas.

Hemos aprendido

A juicio del autor, la planeación tradicional no funciona bien para el desarrollo de software



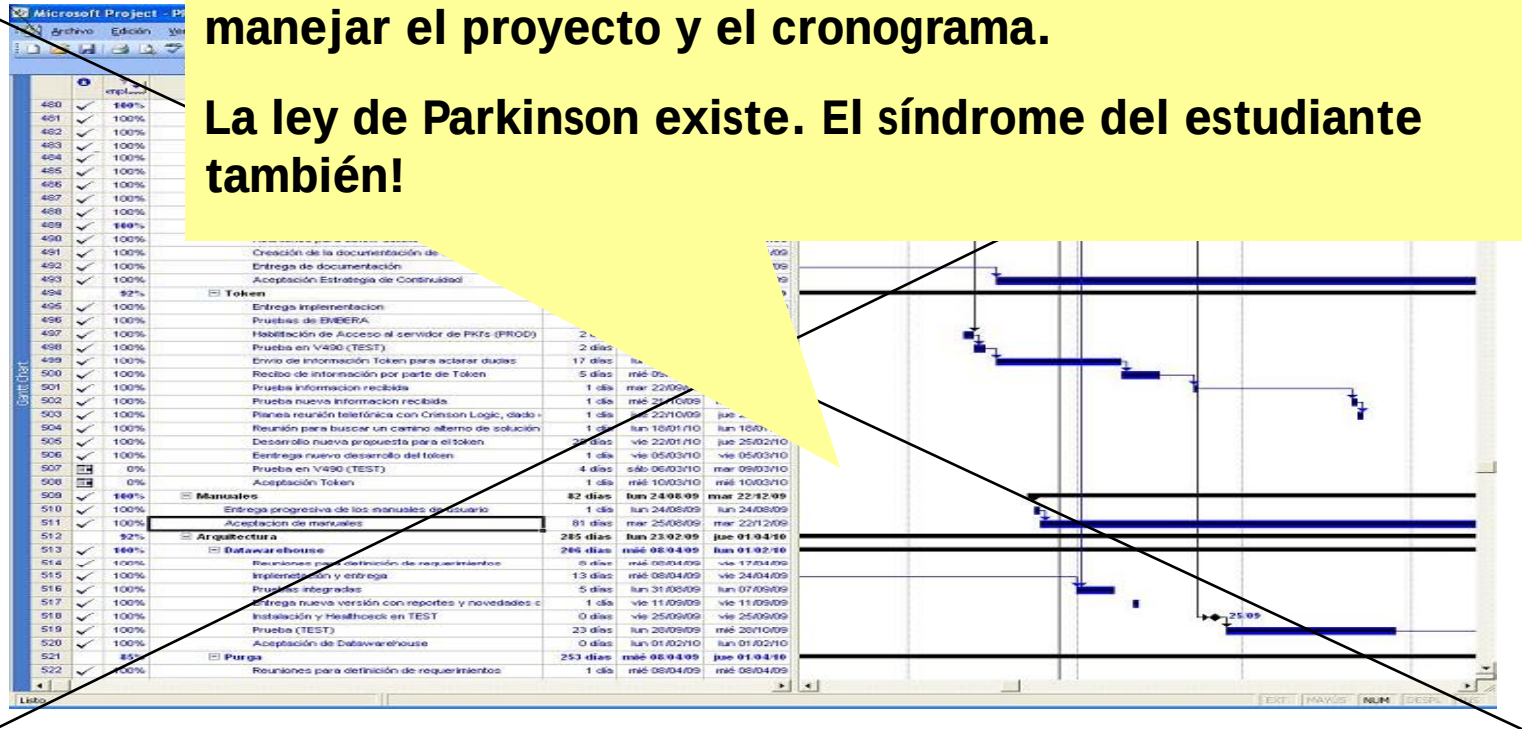
Si va a utilizar diagramas de Gantt, hágalo solo para comunicar. Si está tratando de controlar un proyecto con ellos, está en problemas.

Hemos aprendido

A juicio
bien pa

Si definitivamente *tiene* que utilizar la planeación tradicional, utilice la teoría de restricciones (TOC) para manejar el proyecto y el cronograma.

La ley de Parkinson existe. El síndrome del estudiante también!



Si va a utilizar diagramas de Gantt, hágalo solo para comunicar. Si está tratando de controlar un proyecto con ellos, está en problemas.

Hemos aprendido

A juicio del autor, la especialización de roles, como la entendemos hoy en día, puede ser muy perjudicial para el apropiado desarrollo de un proyecto

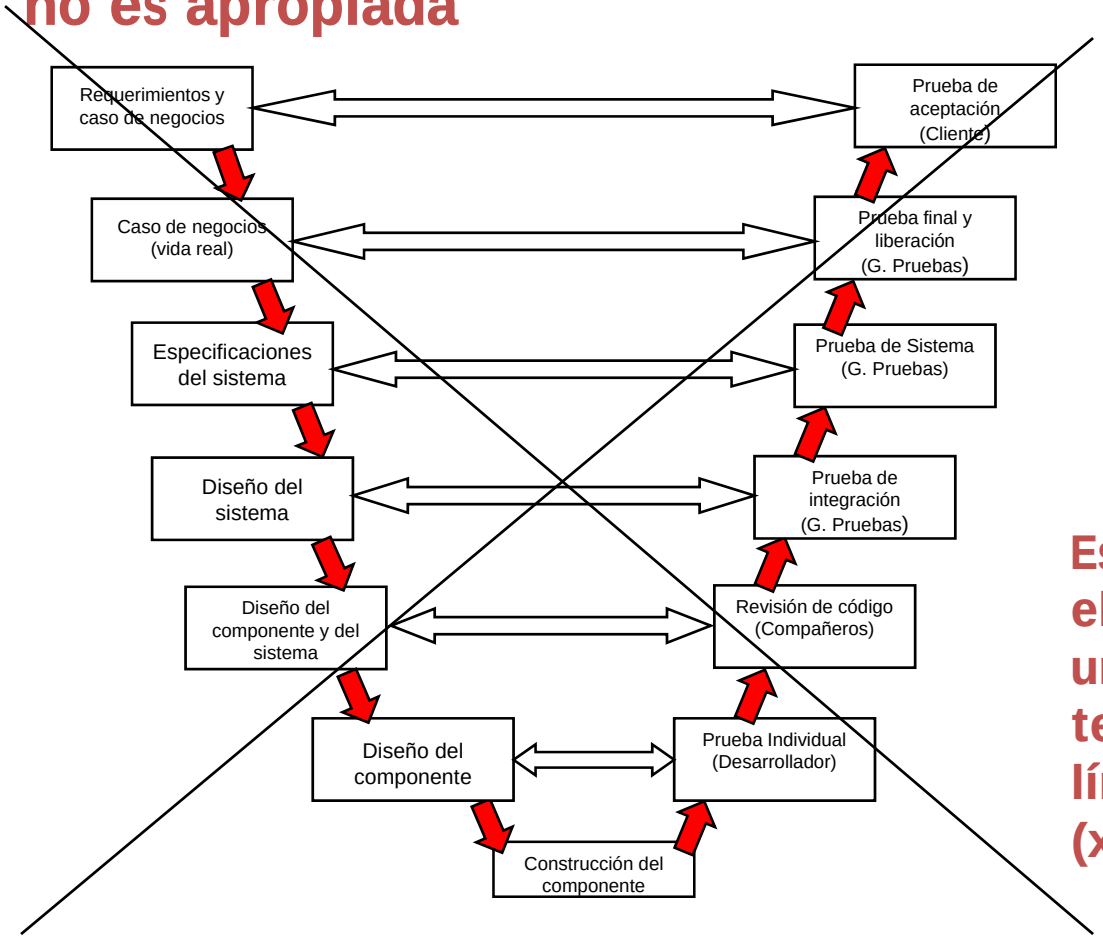
- **Crea dependencias y obliga al desarrollo en cascada**
- **Crea rutas y cadenas críticas (TOC). Al contrario, deben tratar de eliminarse.**
- **Elimina el sentido colectivo que debe tener todo emprendimiento**

Cuando sea posible, *elimine la restricción impuesta por ciertos roles especializados*.

- **Los desarrolladores hacen testing automático**
- **Los desarrolladores capturan requerimientos**
- **Los arquitectos desarrollan**
- **El cliente hace requerimientos y especifica pruebas**

Hemos aprendido

A juicio del autor, la metodología tradicional de pruebas no es apropiada



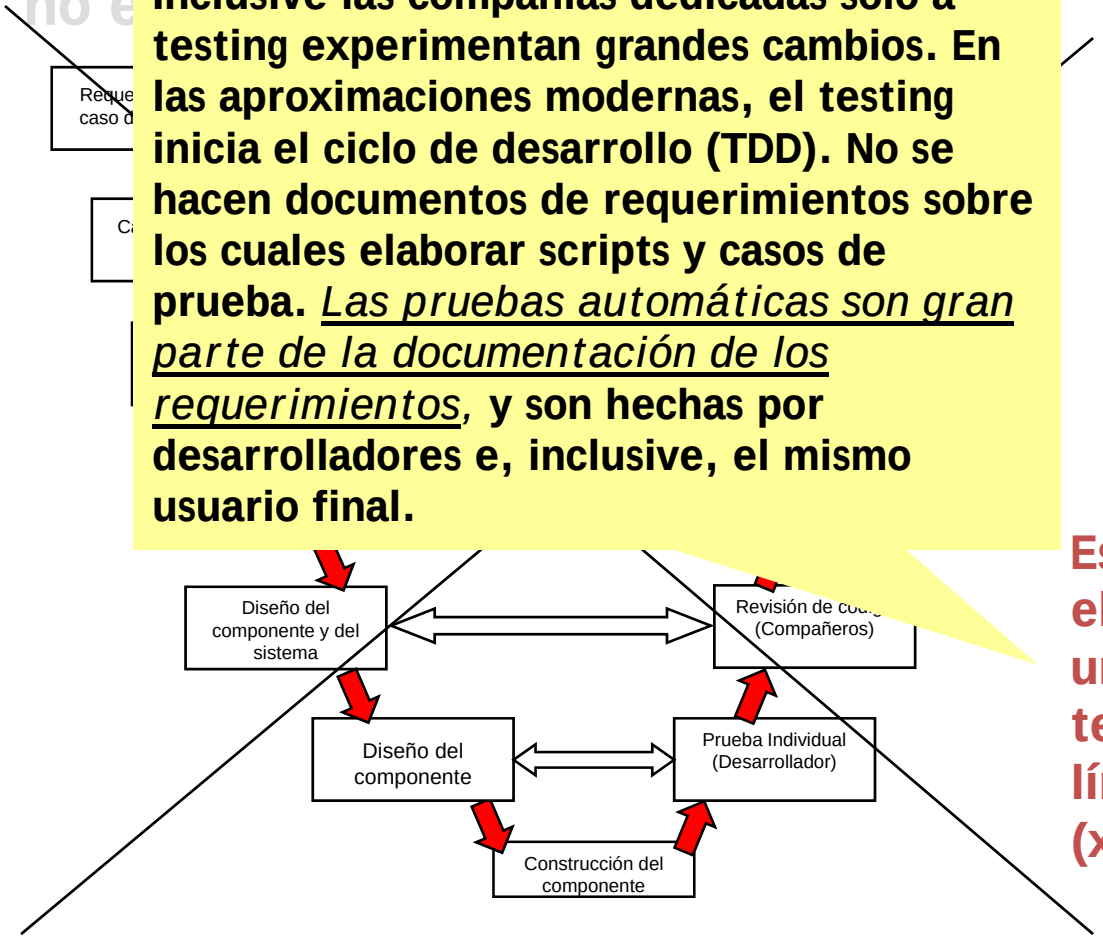
Las pruebas deben ser automáticas, desarrolladas con TDD. Pruebas manuales, solo las estrictamente necesarias.

Es impresionante ver el efecto que en toda una industria han tenido tan pocas líneas de código (xUNIT, FIT, etc.)

Hemos aprendido

A juicio del autor, la metodología tradicional de pruebas no e

Inclusive las compañías dedicadas solo a testing experimentan grandes cambios. En las aproximaciones modernas, el testing inicia el ciclo de desarrollo (TDD). No se hacen documentos de requerimientos sobre los cuales elaborar scripts y casos de prueba. ***Las pruebas automáticas son gran parte de la documentación de los requerimientos,*** **y son hechas por desarrolladores e, inclusive, el mismo usuario final.**



Las pruebas deben ser automáticas, desarrolladas con TDD. Pruebas manuales, solo las estrictamente necesarias.

Es impresionante ver el efecto que en toda una industria han tenido tan pocas líneas de código (xUNIT, FIT, etc.)

Hemos aprendido

A juicio del autor, los proyectos en entidades públicas y en compañías privadas muy ceremoniosas, impersonales y que buscan predictibilidad, seguirán siendo un fracaso

- **En el software no funciona una cultura de comando y control**
- **La participación del cliente es tan importante como la del grupo de ingeniería. No puede por lo tanto desentenderse del problema. Es co – rresponsable.**
- **La complejidad es inherente al problema, no al grupo de ingeniería.**
- **La creación de software no fue, no es y no será prescriptiva ni predictiva. Las licitaciones públicas y los RFP privados de naturaleza similar, son absurdos.**

"Quiero ser ágil… ¿cuando me entrega el cronograma?"

"Quiero ser ágil pero no tenemos tiempo para dedicar al proyecto. Eso es problema suyo"

Hemos aprendido

A juicio del autor, no se puede hacer buen software con personas individualistas, por brillantes que sean

- **En todo emprendimiento exitoso existe una cultura colectiva.**
- **El grupo, no el gerente, puede decidir que no desean trabajar con algún miembro del equipo, incluido el gerente (Scrum Master, Coach, etc.).**
- **Todos ganan o todos pierden. Propiedad colectiva del código.**
- **Si tiene métricas (ojalá las tenga), debe también medir y premiar el desempeño colectivo.**
- **Nunca deje a un desadaptado en un grupo de desarrollo. Nunca!**
- **Quien siempre se defiende de sus errores y no los acepta, es generalmente individualista y sin deseos de aprender.**

Hemos aprendido

A juicio del autor, el lenguaje es extremadamente importante para un ingeniero de software

- **Permite construir y articular conceptos, propósitos y visiones.**
- **Permite comunicar y motivar.**
- **Permite la creación de colectivos.**

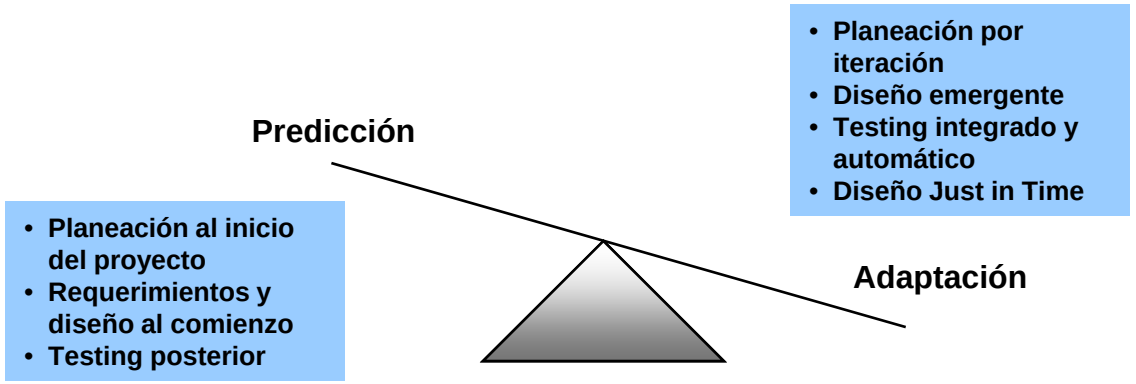
Los propósitos, principios y valores de un grupo de ingeniería, son tan importantes como la destreza técnica

- **Se comprometen.**
- **Cumplen su palabra.**
- **Son honrados**
- **No ocultan sus errores**
- **No mienten**

Hemos aprendido

A juicio del autor, no puede afirmarse que todo lo que emerge de un proyecto ágil, emerge bien

- **Las aproximaciones ágiles favorecen la adaptación frente a la predicción, pero no puede ser una regla. En ocasiones es necesario hacer arquitectura up-front, lo mismo la experiencia de usuario (por lo que son muy criticadas estas aproximaciones).**



¿Y Latinoamérica?

En Latinoamérica enfrentamos los mismos problemas que en las sociedades mas desarrolladas y acceso al mismo conocimiento…

Pero no participamos en la creación de nuevo conocimiento en el campo del desarrollo (¿Ingeniería?) de software.

En las sociedades desarrolladas a los problemas les buscan soluciones

En las sociedades en vías de desarrollo y subdesarrolladas a los problemas les buscamos culpables

"El hombre razonable trata de adaptarse al mundo. El hombre irrazonable persiste en que el mundo se adapte a él. Por lo tanto, todo el progreso depende del hombre irrazonable"

George Bernard Shaw

Muchas gracias