O software e os proxectos de desenvolvemento do software

1. A crise do software

A finais dos anos 50 e principios dos 60, a potencia computacional das máquinas era bastante limitada. Por este motivo, os **programas** que se desenvolvían eran bastante **simples** desde o punto de vista actual. O desenvolvemento de software era, nese momento, unha tarefa artesanal e o seu mantemento era moi custoso. **Programar** era un **arte** para o que se nacía. **Non** existía unha **metodoloxía** ou camiño a seguir para o desenvolvemento de software.

Na década de 1960, o mundo da informática experimentou o que se denominou unha **crise de software** cando os desenvolvedores de software non puideron construír o software que se lles pedía. A demanda de software aumentaba pero a capacidade do software e os desenvolvedores era limitada. Os desenvolvedores non puideron manterse ao día coa complexidade dos proxectos nos que se lles pediu que traballaran. A insto axudou o feito de que a potencia das máquinas aumentou de forma considerable.

Entre as características do software daquel tempo estaban:

- Custes por riba do presupostado
- Retrasos nas entregas
- Prestacións inadecuadas
- Mantemento case imposible
- Modificacións prohibitivas
- Falta de fiabilidade

É posible que agora teñamos máis solucións que nos anos 60, pero estes seguen a ser problemas comúns do software.

A crise do software pode definirse como a dificultade de escribir programas libres de defecto, facilmente comprensibles e que sexan verificables.

As conferencias da OTAN de 1968 e 1969 sobre desenvolvemento de software chegaron á conclusión de que era necesario dar un enfoque de **enxeñaría** no desenvolvemento de software.

A **enxeñaría do software** ten como obxectivos:

- Desenvolver software de calidade
- Aumentar a produtividade
- Diminuír o tempo de desenvolvemento

- Desenvolver software económico

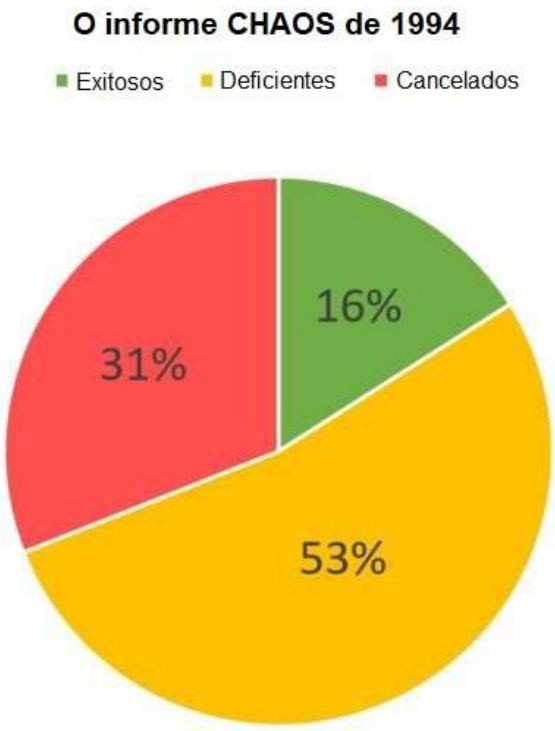
A enxeñaría do software é algo máis que programar. Comeza bastante antes de escribir liñas de código e continúa despois de que se ten completado a primeira versión do produto.

O proceso, a administración, a planificación, o seguimento e o control rigoroso do proxecto é esencial.

Podemos dicir que a crise do software se trata máis dunha **enfermidade crónica** que dunha crise puntual.

Existen varios estudos ou estatísticas sobre o estado dos proxectos de software, sendo o máis utilizado o informe de Standish Group, chamado **Chaos Report**. Este informe é a estatística de referencia máis citada en enxeñaría do software. Entre estes informes Chaos, o máis famoso é o publicado en 1994, e que de modo resumido informaba de que:

- 31% dos proxectos se cancelaron.
- 53% tiñan deficiencias.
- 16% foron un éxito.
- De media os proxectos tiñan un 189% de sobrecustes.
- De media os proxectos tardaban en realizarse un 222% sobre o tempo orixinal estimado.



No ano 2015 o informe CHAOS analizou 50.000 proxectos en todo o mundo, desde pequenas melloras ata implementacións masivas de reenxeñaría de sistemas.

Os resultados amosan que aínda queda traballo por facer para lograr resultados satisfactorios nos proxectos de software.

EVOLUCIÓN DOS INFORMES CHAOS

	1994	2000	2006	2012	2015
Exitosos	16%	28%	35%	27%	29%
Deficientes	53%	49%	46%	56%	52%
Fracasos	31%	23%	19%	17%	19%

O informe mide o éxito dos proxectos só en base a se remataron en **tempo**, **presuposto** e cumpriron cos **requisitos**, deixando fora aspectos como a calidade, o risco e a satisfacción do cliente.

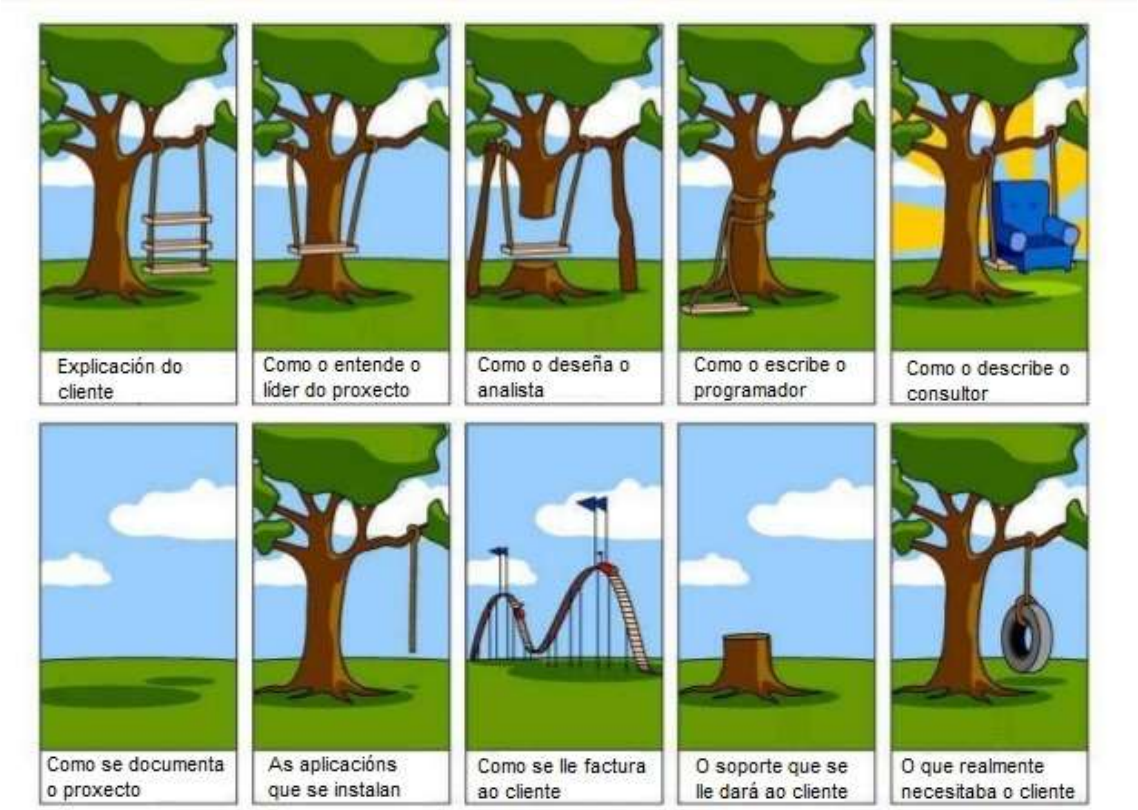
Buscando algún patrón para determinar como maximizar o éxito dos proxectos e minimizar o fracaso, dentro do informe CHAOS amósanse estas porcentaxes segmentadas polo tamaño dos proxectos. O resultado é claro. Un proxecto **pequeno** ten máis probabilidades de éxito.

INFORME CHAOS SOBRE O TAMAÑO DOS PROXECTOS

	ÉXITO	DEFICIENTE	FRACASO
Enorme	2%	7%	17%
Grande	6%	17%	24%
Medio	9%	26%	31%
Moderado	21%	32%	17%
Pequeno	62%	16%	11%
TOTAL	100%	100%	100%

O informe mostra resultados de proxectos realizados entre os anos 2011 e 2015

Na seguinte imaxe, móstrase a gran problemática do desenvolvemento de software para un proxecto de construción dun columpio.



2. O ciclo de vida do software

O ciclo de vida do software é unha sucesión de **etapas** polas que pasa o software no seu desenvolvemento, desde que se concibe a idea ata que o software deixa de utilizarse (obsolescencia).

O ciclo de vida dun proxecto especifica o enfoque xeral do desenvolvemento, indicando os procesos, actividades e tarefas que se van realizar e en que orde, e os produtos que se van xerar, os que se van entregar ao cliente e en que orde se van entregar.

Un **proceso** é un conxunto de actividades que se suceden seguindo unha ordenación temporal determinada.

Unha **actividade** é un conxunto de tarefas.

Unha **tarefa** é unha acción que transforma entradas en saídas.

2.1. Modelo de ciclo de vida en cascada

O modelo en cascada (waterfall model en inglés) foi un dos primeiros modelos de ciclo de vida que formalizou un conxunto de procesos de desenvolvemento de software.

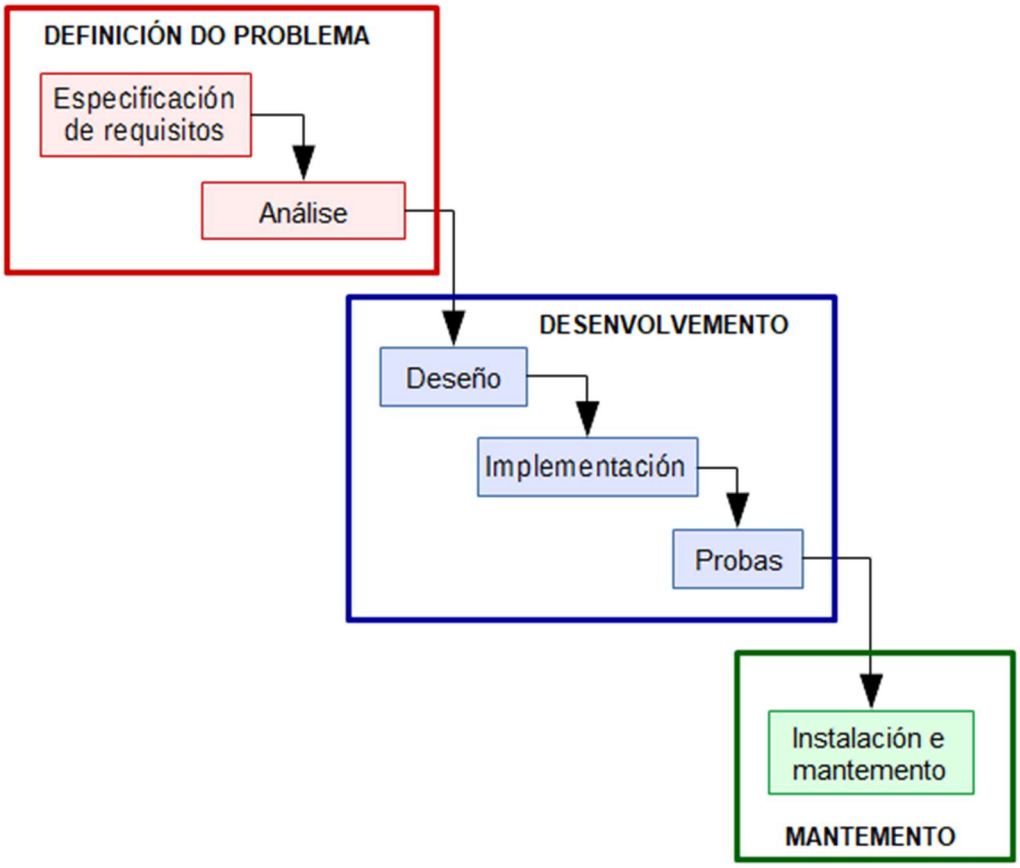
Neste modelo as etapas realízanse unha detrás da outra de forma lineal. Só cando a primeira fase remata, pode comezar a segunda e así sucesivamente.

Este modelo asume que todo se leva a cabo e ten lugar tal e como se planeara na fase anterior.

Este modelo é recomendable cando o desenvolvedor xa ten deseñado e desenvolvido aplicacións similares con anterioridade, é dicir, ten a experiencia suficiente para rematar cunha etapa e comezar coa seguinte.

Podemos agrupar en **tres fases** as etapas deste ciclo de vida:

- **Definición do problema**. Inclúe tanto a especificación de requisitos como a análise do sistema.
- **Desenvolvemento**. Abarca o deseño, a implementación e as probas do sistema.
- **Mantemento**. Inclúe a instalación e mantemento do sistema.



Especificación de requisitos. Un requirimento é unha característica do sistema ou unha descrición de algo que o sistema é capaz de facer co obxecto de satisfacer o propósito do sistema. Os requirimentos son o que os clientes/usuarios esperan que faga o sistema. Os analistas, polo tanto, deben

entender o problema dos usuarios na súa cultura e coa súa linguaxe e construír o sistema que resolve as súas necesidades. Para isto recóllense todos os requisitos no documento de especificación de requisitos do produto.

Nesta fase poden xurdir moitos problemas, desde un sistema complexo cunha tecnoloxía que non domina o analista, ata que o cliente non sabe con exactitude o que realmente quere.

Análise. Nesta fase defínense esquemas, modelos e regras de negocio en función dos requisitos obtidos na fase anterior. Os documentos xerados revísanse co cliente para obter a súa aprobación. Esta etapa é de grande importancia xa que nela se basea todo o desenvolvemento posterior. Ademais, a expresión formal das especificacións soe ter un carácter contractual.

Deseño. O obxectivo da fase de deseño é transformar os requisitos especificados na fase de análise nunha estrutura adecuada para a implementación nalgunha linguaxe de programación.

Implementación. Na fase de codificación, o deseño do software tradúcese a código fonte utilizando calquera linguaxe de programación adecuada.

Probas. Nesta fase próbase a aplicación para tentar atopar defectos. Os defectos tentan corrixirse e volven facerse probas para ver que os mencionados defectos desapareceron.

Mantemento. O mantemento é a fase máis importante do ciclo de vida do software. O esforzo dedicado ao mentemento é o 60% do esforzo total dedicado a desenvolver un software completo. Basicamente hai tres tipos de mantemento:

- **Mantemento correctivo**. Este tipo de mantemento realízase para corrixir erros que non se descubriron durante a fase de desenvolvemento do produto.
- **Mantemento perfectivo**. Este tipo de mantemento lévase a cabo para mellorar as funcionalidades do sistema en función das solicitudes do cliente.
- **Mantemento adaptativo**. O mantemento adaptativo xeneralmente requírese para que o software funcione nun novo contorno, como o traballo nunha nova plataforma informática ou cun novo sistema operativo.

Este modelo non acepta cambios e o produto só se obtén ao final do proceso polo que pode pasar moito tempo entre a proposta inicial e a entrega do sistema.

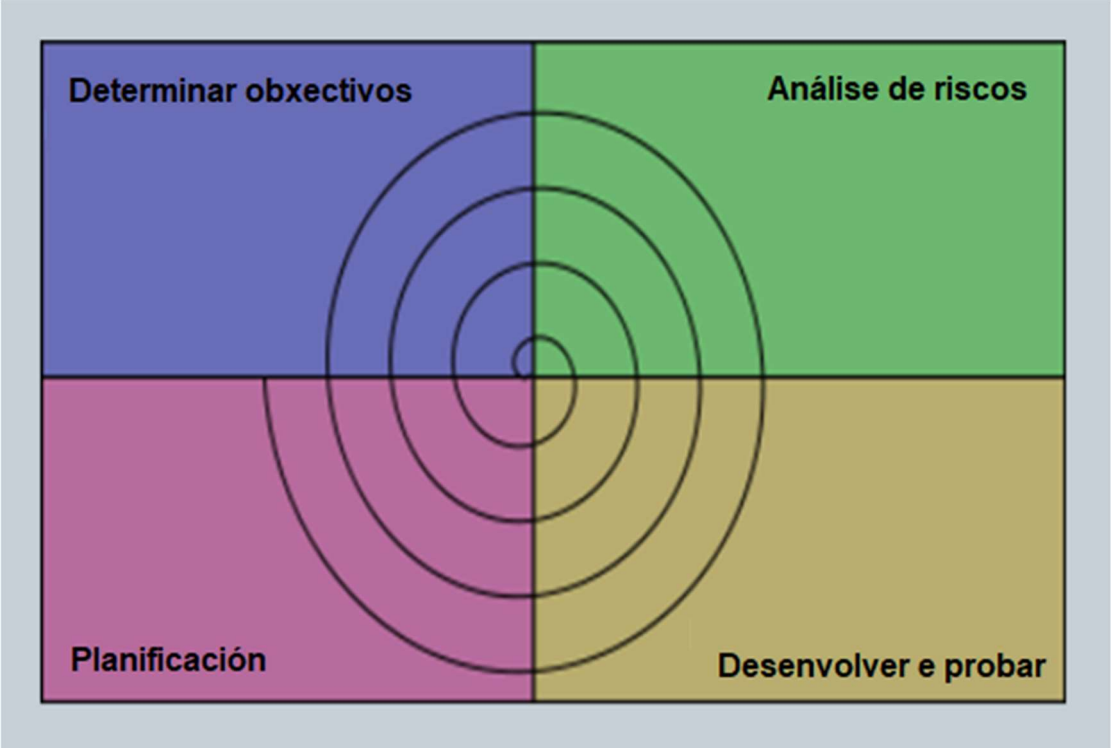
É recomendable para proxectos curtos , simples e directos. Non é bo se hai cambios nos requirimentos ou riscos posibles.

2.2. Modelo de ciclo de vida en espiral

O modelo en espiral está orientado ao risco, onde en cada ciclo se avalía o risco e continúa cun ciclo normal (cascada).

O modelo de ciclo de vida en espiral, consiste en realizar diversas iteracións, pasando por cada unha das súas fases unha e outra vez. Baséase, por tanto, nunha serie de ciclos repetitivos para ir gañando madurez ata chegar ao produto final.

Proporciona o potencial para o desenvolvemento rápido de versións incrementais do software que non se basea en fases claramente definidas e separadas para crear un sistema.



No modelo en espiral orixinal non hai un número definido de iteracións. As iteracións debe decidilas o equipo de xestión.

Cada volta divídese en catro sectores:

Definición de obxectivos. Nesta fase do proxecto defínense os obxectivos específicos. Identifícanse as restricións do proceso e o produto e estipúlase un plan detallado de administración. Identifícanse os riscos, planifícanse estratexias alternativas.

Análise de riscos. Lévase a cabo unha análise detallada de cada un dos riscos do proxectos. Defínense os pasos para reducir eses riscos.

Desenvolvemento e probas. Unha vez elixida a alternativa que se vai construír, empréndese o traballo de levala a cabo, é dicir, crear o produto, crear o proxecto, etc.

Planificación. Revísase o proxecto e tómase a decisión de seguir ou non cun ciclo posterior da espiral. No caso de decidir continuar, desenvólvense os plans para a seguinte fase do proxecto.

Con cada iteración ao redor da espiral (comezando no centro e seguindo cara ao exterior), constrúense sucesivas versións do software, cada vez máis completas e, ao final, o propio sistema software totalmente funcional.

2.3. Ciclo de vida adaptativo ou áxil

Os ciclos de vida adaptativos, tamén coñecidos como métodos orientados ao cambio ou métodos áxiles, responden a niveis altos de cambio e á participación continua dos interesados.

Existen dous modelos básicos para este tipo de ciclos de vida, aqueles centrados no fluxo (por exemplo **Kanban**) e outros centrados en ciclos iterativos e incrementais (por exemplo **Scrum**).

Os principios áxiles son os seguintes:

- Colaboración estreita co cliente
- Predisposición e resposta ao cambio
- Desenvolvemento incremental con entregas frecuentes de funcionalidade
- Comunicación verbal directa
- Simplicidade, só se constrúen os artefactos necesarios
- Motivación, compromiso e responsabilidade do equipo pola autoxestión, autoorganización

2.3.1. Scrum

Scrum é un **marco de traballo** para o desenvolvemento de software. En lugar de proporcionar unha descrición completa e detallada de como deben realizarse as tarefas dun proxecto, deixa moito en mans do equipo de desenvolvemento. Isto pasa debido a que é o equipo quen coñecerá a mellor maneira de resolver as problemáticas que se presentan.

O equipo de desenvolvemento apóiase en **dous roles**: o Scrum Master e o Product Owner. O **Scrum Master** é quen vela pola produtividade do equipo de desenvolvemento. Pode considerarse un coach ou líder facilitador encargado de acompañar ao equipo de desenvolvemento de forma que atope o seu punto de maior eficiencia. O **Product Owner** é quen representa ao negocio, os interesados (stakeholders) no proxectos, o cliente e os usuarios finais. Ten a responsabilidade de conducir ao equipo de desenvolvemento cara ao produto adecuado.

O progreso dos proxectos que utilizan Scrum realízase e verifica nunha serie de iteracións chamadas **Sprints**. Estes Sprints teñen una duración fixa, preestablecida de non máis dun mes. Ao comezo de cada Sprint o equipo de desenvolvemento realiza un compromiso de entrega dunha serie de funcionalidades ou características do produto en cuestión.

Ao finalizar o Sprint espérase que estas características comprometidas estean rematadas, o que implica a súa análise, deseño, desenvolvemento, proba e integración ao produto en desenvolvemento. Neste momento é cando se realiza unha reunión de revisión do produto construído durante o Sprint, onde o equipo de desenvolvemento amosa o construído ao Product Owner e a calquera stakeholder interesado en participar. A realimentación obtida nesta reunión pode incluírse entre as funcionalidades a construír en futuros Sprints.

Vídeo explicativo: https://www.youtube.com/watch?v=P25JP0u6UKw

Máis información:

http://media.kleer.la/kleer-introduccion-a-agile-scrum-es.pdf

https://www.scrummanager.net/files/scrum_manager.pdf

2.3.2. Kanban

Kanban é unha palabra xaponesa que significa algo así como “tarxetas visuais” (kan significa visual e ban tarxeta). Esta técnica creouse en Toyota e utilízase para controlar o avance do traballo, no contexto dunha liña de produción.

Kanban non é unha técnica específica de desenvolvemento de software, o seu obxectivo é xestionar de maneira xeral como se van completando as tarefas, pero nos últimos anos tense utilizada na xestión de proxectos de desenvolvemento de software, a miúdo con Scrum.

As principais regras de Kanban son as tres seguintes:

- Visualizar o traballo e as fases do ciclo de produción ou fluxo de traballo

- Determinar o límite do traballo en curso
- Medir o tempo en completar unha tarefa

Ao igual que Scrum, Kanban baséase no desenvolvemento incremental, dividindo o traballo en partes. Unha das principais aportacións é que utiliza técnicas visuais para ver a situación de cada tarefa.

O traballo divídese en partes. Normalmente, cada unha destas partes escríbese nun post-it e pégase nun muro ou tableiro. Os post-it soen ter información variada, pero, ademais da descrición, deberían ter unha estimación da duración da tarefas.

O muro ten tantas columnas como estados polos que pode pasar unha tarefa (exemplo, en espera de ser desenvolvida, en análise, en deseño, etc.

O obxectivo desta visualización é que quede claro o traballo a realizar, en que está traballando cada persoa, que todo o equipo teña algo que facer e ter claras as prioridades das tarefas. As fases do ciclo de produción ou fluxo de traballo decídense segundo o caso, non hai nada acotado.

Quizais unha das principais ideas de Kanban é que o traballo en curso debería estar limitado, é dicir, que o número máximo de tarefas que se poden realizar en cada fase debe ser algo coñecido.

En Kanban debe definirse cantas tarefas como máximo poden realizarse en cada fase do ciclo de traballo (exemplo, como máximo 4 tarefas en desenvolvemento, como máximo 1 en probas, etc). A este número de tarefas chámaselle límite do

“work in progress”. A isto engádeselle outra idea razoable como que para comezar cunha nova tarefa algunha outra tarefa previa debe ter finalizado.

Vídeo comparativo entre Scrum e Kanban:

https://www.youtube.com/watch?v=8qa6io8wHQA