Gestión Desarrollo de Software – 2025

Prof. Politi Raul

Clase 19 (Unidad 6)

Distribución de tiempos para entregables intermedios y finales

Objetivo: Proveer un marco práctico y metodológico para dividir el proyecto en entregables intermedios y finales, establecer fechas realistas, definir hitos (milestones) y asegurar entregas con calidad y valor para el cliente.

1. Definición de entregables y hitos

Un entregable (deliverable) es un resultado tangible o verificable que debe entregarse a lo largo del proyecto. Los hitos (milestones) son puntos en el tiempo que marcan la finalización de una fase, entrega o evento importante.

Tipos de entregables:

- - Entregables intermedios: prototipos, versiones alfa/beta, documentación técnica, pruebas de integración.
- - Entregables finales: versión de producción, manual del usuario, despliegue en entorno productivo.

2. Principios para distribuir tiempos

1. Descomposición: dividir el proyecto en paquetes de trabajo (WBS) hasta llegar a tareas que se puedan estimar con confianza (idealmente 1–5 días de trabajo).

2. Priorización por valor: priorizar entregables que aporten mayor valor al usuario o que reduzcan mayor incertidumbre (MVP-first).

3. Dependencias explícitas: identificar dependencias entre tareas y entregables para ordenar cronológicamente las entregas.

4. Buffers y contingencias: reservar tiempo para imprevistos (10–30% según riesgo), integración y pruebas.

5. Validación temprana: planificar entregables que permitan feedback temprano del cliente y ajuste de alcance.

3. Técnicas de estimación aplicadas a entregables

A continuación las técnicas más útiles para estimar tiempo de entregables:

- - Estimación por analogía:

- Comparar con proyectos previos similares y ajustar por diferencias (complejidad, equipo, tecnología).
- - Descomposición (bottom-up):
- Estimar tareas pequeñas y sumar para obtener estimación del entregable. Mayor precisión, pero consume tiempo.
- - PERT (Program Evaluation and Review Technique):
- Uso de tres estimaciones: optimista (O), más probable (M) y pesimista (P). Fórmula de la duración esperada: (O + 4M + P)/6. Útil cuando hay incertidumbre.
- - Planning Poker / Estimación por puntos de historia:
- Técnica colaborativa para estimar esfuerzo relativo usando puntos (Fibonacci). Se usa en equipo ágil.

4. Planificación de hitos y calendario

Para crear un calendario de entregables:
1) Definir lista de entregables y criterios de aceptación (DoD - Definition of Done).
2) Relacionar entregables a paquetes de trabajo y estimar duración.
3) Identificar dependencias y calcular ruta crítica.
4) Añadir buffers por riesgo e integración.
5) Revisar con stakeholders y acordar fechas.

- Ejemplo de hitos (cronograma de alto nivel):
- - Hito 1: Entrega prototipo funcional (Fecha)
- - Hito 2: Entrega versión beta / pruebas internas (Fecha)
- - Hito 3: Entrega candidata a producción (Release Candidate) (Fecha)
- - Hito 4: Entrega final / puesta en producción (Fecha)

5. Control y seguimiento de entregables

Métricas y artefactos para controlar entregables:

- - Burn-down / Burn-up charts para seguimiento del progreso.
- - KPI: cumplimiento de hitos, porcentaje de entregables completados, desviación de tiempo.
- - Reuniones de revisión: demo de entregables, retrospective y reuniones de seguimiento con stakeholders.

6. Gestión de riesgos en la planificación de entregables

Incluir un registro de riesgos asociado a entregables: identificar riesgos, probabilidad, impacto, plan de mitigación y propietario del riesgo. Ajustar tiempos según riesgo residual.

7. Plantilla práctica: cómo definir un entregable

Cada entregable debe incluir:

- - Identificador y nombre
- - Descripción breve
- - Criterios de aceptación (DoD)

- - Dependencias
- - Estimación (horas/días)
- - Recursos asignados
- - Fecha objetivo y riesgo asociado

8. Buenas prácticas y recomendaciones

- Entregá valor temprano y frecuentemente: reduce riesgo y aumenta feedback.
- Mantén reuniones cortas de revisión con stakeholders; valida hipótesis y ajusta priorizaciones.
- No sobrecargues sprints/iteraciones; respeta la capacidad real del equipo.
- Documenta supuestos y versiones para poder justificar cambios de fechas.

Definición y planificación de iteraciones en un proyecto de software

Objetivo: Explicar cómo definir iteraciones (sprints), planificarlas, estimar su contenido, gestionar backlog, medir velocidad y ajustar expectativas de entrega.

1. Qué es una iteración

Una iteración (o sprint en Scrum) es un periodo de tiempo fijo (por ejemplo 1–4 semanas) durante el cual el equipo entrega un incremento de producto potencialmente utilizable. Las iteraciones permiten entrega continua de valor y adaptabilidad.

2. Roles y artefactos relacionados

- - Product Owner: prioriza el backlog y define el valor del negocio.
- - Scrum Master / Facilitador: quita impedimentos y protege al equipo.
- - Equipo de desarrollo: implementa y prueba las historias.
- - Artefactos: Product Backlog, Sprint Backlog, Increment, Definition of Done (DoD).

3. Ciclo de una iteración

1) Planificación de iteración (Sprint Planning): seleccionar historias del backlog según prioridad y capacidad.
2) Ejecución: desarrollo, integración continua, pruebas.
3) Daily Stand-up: comunicación diaria y remover impedimentos.
4) Review / Demo: mostrar el incremento al Product Owner y stakeholders.
5) Retrospective: analizar qué mejorar para la próxima iteración.

4. Planificación: cómo decidir qué entra en la iteración

Pasos prácticos:

- - Refinamiento del backlog: dividir historias grandes (epics) en historias pequeñas (ideal ≤ 5 días).
- - Estimación por puntos de historia o tiempo: usar técnicas como Planning Poker para consenso del equipo.
- - Calcular la capacidad del sprint: considerando feriados, vacaciones y otros compromisos.

- - Seleccionar historias hasta cubrir la capacidad estimada (no sobrepasar).

5. Estimación y métricas

Métricas comunes:

- - Velocidad (velocity): puntos completados por iteración. Se usa para predecir entregas futuras.
- - Burn-down chart: muestra trabajo restante en la iteración.
- - Cycle time / Lead time: tiempo desde que una historia entra hasta que se entrega.
- - Throughput: número de ítems completados por período.

6. Dependencias, riesgos y límites WIP

Identificar dependencias entre historias que puedan bloquear el progreso. Limitar trabajo en progreso (WIP) para evitar multitasking y pérdida de eficiencia. Gestionar riesgos con mitigaciones y contingencias dentro de la iteración.

7. Técnicas prácticas durante la iteración

- - Integración continua y despliegue continuo (CI/CD): automatizar builds y tests para reducir riesgo de integración.
- - Test-driven development (TDD) / pruebas automatizadas: asegurar calidad desde el inicio.
- - Pair programming / Code review: mejorar calidad y compartir conocimiento.

8. Ejemplo de planificación de iteración (plantilla)

Supongamos sprint de 2 semanas (10 días hábiles). Capacidad del equipo = 40 puntos (por ejemplo).

Paso a paso:

- 1) Refinar backlog y estimar historias (Planning Poker).
- 2) Seleccionar historias por prioridad hasta sumar ~40 puntos.
- 3) Asignar tareas diarias y estimaciones en horas para seguimiento.
- 4) Monitorizar burn-down diario y resolver impedimentos.

Plantilla mínima para cada historia:

- - ID / Título
- - Descripción breve
- - Criterios de aceptación (DoD)
- - Puntos estimados
- - Dependencias
- - Responsable

9. Planificación a largo plazo: Release Planning

A partir de la velocidad y backlog priorizado, es posible planificar releases (entregas mayores) que agrupen varias iteraciones. Se debe tener en cuenta riesgos, descubrimientos y buffers para asegurar fechas de entrega realistas.

10. Buenas prácticas y anti-patrones

Buenas prácticas:

- - Mantener historias pequeñas y verticales (end-to-end).
- - Involucrar al Product Owner en la priorización y aceptación.
- - Revisar constantemente la capacidad y ajustar la planificación.

Anti-patrones:

- - Sobrecargar el sprint con demasiado alcance.
- - No refinamiento del backlog (historias grandes / vagamente definidas).
- - Ignorar feedback temprano del cliente.

11. Ejercicios sugeridos para la clase

- - Ejercicio 1: Dado un backlog y capacidades, realizar la planificación del sprint y justificar la selección de historias.
- - Ejercicio 2: Calcular la velocidad promedio a partir de tres sprints y estimar cuántos sprints hacen falta para una release.
- - Ejercicio 3: Crear un burn-down chart ficticio y analizar causas de desviación.

12. Resumen

Las entregas intermedias y la planificación de iteraciones son prácticas complementarias que permiten entregar valor de forma incremental, mitigar riesgos y ajustar expectativas de forma continua. Una buena práctica combina descomposición, estimaciones apropiadas, feedback temprano y control de riesgos.