Crystal Clear y Desarrollo esbelto de software (DES)
Travelix

Daniel Toledo Daniel Machado Osvaldo Moreno José Antonio Concepción
C-311 - Agencia de Viajes. Equipo 3 Semigrupo 2.

20 de febrero de 2024

Índice

1. Información teórica sobre la metodología en la literatura.

Crystal Clear

Contexto y Fundamentos: Crystal Clear es parte de la familia de métodos ágiles creada por Alistair Cockburn y Jim Highsmith. Estos métodos están diseñados para abordar el desarrollo de software como un "juego cooperativo con recursos limitados, de invención y comunicación". El enfoque principal es lograr la "maniobrabilidad", entregando software útil y funcional. Crystal Clear es uno de los ejemplos dentro de la familia Crystal, que comparte un "código genético" común pero se adapta a las características específicas de cada proyecto. Cockburn y Highsmith han definido un conjunto de metodologías dentro de esta familia, cada una con elementos fundamentales comunes y componentes únicos.

Elementos Comunes:

- **Roles:** Se establecen roles específicos para los miembros del equipo, cada uno con responsabilidades claras.
- **Patrones de Proceso:** Existen patrones de proceso que son compartidos entre las metodologías de la familia Crystal, proporcionando una base consistente.
- **Producto del Trabajo:** Se define qué se espera como resultado del trabajo, manteniendo consistencia a través de los diferentes métodos.

La maniobrabilidad es clave en Crystal Clear, permitiendo a los equipos ágiles seleccionar el miembro de la familia Crystal más apropiado para su proyecto y entorno. Esta flexibilidad se traduce en la capacidad de adaptarse a diferentes tipos de proyectos.

Desarrollo Esbelto de Software (DES)

El Desarrollo Esbelto de Software (DES) adapta los principios de la manufactura esbelta al ámbito de la ingeniería de software. Los principios de esbeltez que guían el proceso DES se resumen de la siguiente manera:

1. **Eliminar el desperdicio:**
   - No agregar características o funciones innecesarias.
   - Evaluar el costo y el impacto de nuevos requerimientos.
   - Eliminar etapas superfluas del proceso.
   - Mejorar la obtención de información por parte del equipo.
2. **Generar calidad:**
   - Asegurarse de que las pruebas detecten tantos errores como sea posible.
   - Mejora continua de la calidad del software.
3. **Crear conocimiento:**
   - Fomentar la generación y acumulación de conocimiento.
   - Promover la colaboración y el intercambio de información entre el equipo.

4. **Aplazar el compromiso:**
    - Reducir el tiempo necesario para decisiones que afectan al software o al proceso.
    - Mantener la flexibilidad para adaptarse a cambios en los requisitos.
5. **Entregar rápido:**
    - Buscar la entrega temprana y continua de versiones funcionales del software.
    - Incrementar la retroalimentación rápida del cliente.
6. **Respetar a las personas:**
    - Promover un entorno de trabajo colaborativo y respetuoso.
    - Reconocer y valorar las habilidades individuales del equipo.
7. **Optimizar al todo:**
    - Simplificar la transmisión de información a todos los participantes.
    - Buscar la optimización global del proceso.

2. Aplicación de los exponentes.

Como hemos aclarado el DES se enfoca en eliminar el desperdicio, generar calidad, crear conocimiento, aplazar el compromiso, entregar rápidamente, respetar a las personas y optimizar el proceso en su totalidad. Para poder aplicar estos principios en el contexto del problema de la agencia de viajes podríamos realizar las siguientes acciones.

- **¿Cómo eliminar el desperdicio?**: Mediante la evaluación cuidadosa de los requerimientos del cliente se evitará agregar características innecesarias al producto. Se establecerán tareas concisas y claras para que se pueda 'saltar' directamente a sus implementaciones y no extenderse innecesariamente en los análisis previos.
- **Ciclos de trabajo:** Buscaríamos la entrega temprana y continua de versiones funcionales del software para obtener retroalimentación rápida del cliente y permitir ajustes o mejoras rápidas. Se podrían establecer ciclos de desarrollo cortos (10 y 14 días dependiendo de la complejidad de la funcionalidad) y realizar demostraciones regulares al cliente para obtener feedback.
- **¿Cómo generar calidad?**: Un integrante del equipo se centraría en la realización de testers para detectar errores tempranamente, para que este proceso tenga un mejor resultado el creador del test debe ser diferente del desarrollador de la funcionalidad, así se puede mejorar continuamente la calidad del software mediante la retroalimentación entre los integrantes del equipo de trabajo, además de la interacción con el cliente ya que este se mantiene disponible a nuestras inquietudes. Se garantiza la robustez porque en cada uno de los ciclos cortos las nuevas implementaciones se someten a un proceso de pruebas intensivo, garantizando la detección temprana de errores y sus soluciones, pudiendo entregar una versión estable del software.

- **¿Cómo fomentar el conocimiento de todos?**: Aplicando la colaboración entre los miembros del equipo y promoviendo el intercambio de información y experiencia. Se podrían realizar sesiones periódicas de retroalimentación y aprendizaje para compartir conocimientos y mejorar las habilidades del equipo. Estas reuniones deben realizarse cómo máximo una vez en la semana de trabajo, priorizando la realización de la documentación de cada funcionalidad en el momento de su finalización y ahorrar tiempo a la larga.

3. Análisis crítico del exponente.

3.1. Ventajas y desventajas.

1. Entre las **ventajas** de Crystal Clear se encuentran:
   - Enfoque en equipos pequeños y altamente colaborativos.
   - Comunicación efectiva y colaboración entre los miembros del equipo.
   - Entrega temprana y frecuente de software funcional.
   - Adaptabilidad a las necesidades específicas del proyecto y del equipo.
   - Enfoque en la simplicidad y la flexibilidad.
2. Sin embargo, también existen algunas **desventajas**, como:
   - Puede ser difícil de implementar en equipos grandes o en proyectos complejos.
   - Requiere un alto nivel de compromiso y colaboración por parte de todos los miembros del equipo.
   - Puede ser difícil de seguir para equipos sin experiencia en metodologías ágiles.

3.2. Roles y su importancia.

El rol del **jefe de equipo** en el desarrollo de software puede ser crucial para el éxito de un proyecto. Un buen jefe de equipo puede:

- Definir la dirección estratégica del proyecto y mantener al equipo alineado con los objetivos.
- Desarrollar un plan de proyecto viable, asignar tareas y gestionar el tiempo y los recursos de manera eficiente.
- Inspirar y motivar al equipo, fomentar la colaboración y resolver conflictos.
- Tomar decisiones difíciles de manera oportuna y responsable, especialmente en situaciones de incertidumbre.
- Facilitar la comunicación clara y abierta entre los miembros del equipo, los superiores y el cliente.

Autocontrol vs. liderazgo:
El autocontrol es una habilidad importante para los miembros del equipo, pero no sustituye el liderazgo. Un equipo auto organizado puede funcionar bien en algunos casos, pero sin un líder fuerte, puede enfrentar desafíos como:

- Perder el foco o desviarse de los objetivos principales.

- Puede ser difícil determinar quién es responsable de qué tareas o decisiones.
- El proceso de consenso puede tomar demasiado tiempo, especialmente en situaciones críticas.
- Los conflictos pueden escalar sin la intervención de un mediador imparcial.

Influencias y preferencias personales:
Incluso en un equipo auto organizado, las decisiones no estarán completamente libres de influencias y preferencias personales. Es natural que los miembros del equipo tengan diferentes opiniones y prioridades, y el jefe de equipo juega un papel crucial en:

- Asegurar que las decisiones se tomen en base al mejor interés del proyecto y no en favor de individuos o grupos.
- Ayudar a los miembros del equipo a manejar sus emociones y evitar que afecten el trabajo.
- Fomentar un ambiente donde se valoren todas las opiniones y se consideren diferentes perspectivas.

En el caso de nuestro equipo, tenemos la ventaja de que hemos trabajado juntos con anterioridad lo que contribuye a un mejor funcionamiento y flujo de trabajo. Tenemos un buen jefe de equipo y además somos capaces de autogestionarnos con orientaciones mínimas de las tareas a realizar, lo que posibilita una excelente sinergia en términos productivos. En este aspecto consideramos esencial la buena comunicación, la disciplina y la seriedad con el cumplimiento de los objetivos de trabajo.

3.3. ¿Equipos auto organizados conduce a "Las mejores arquitecturas, requisitos y diseños"?

Desde un punto de vista positivo, la autoorganización fomenta la participación activa de todos los miembros del equipo en la toma de decisiones, lo que puede resultar en una mayor diversidad de ideas y enfoques, así como en una mayor satisfacción y compromiso por parte de los miembros del equipo. Esto, a su vez, puede llevar a la identificación de soluciones más innovadoras y creativas, así como a la adaptación ágil a los cambios en los requisitos del proyecto. Sin embargo, desde una perspectiva más crítica, la autoorganización también puede presentar desafíos, especialmente en equipos sin la experiencia o la capacitación adecuadas en metodologías ágiles. La falta de liderazgo claro y la ausencia de roles definidos pueden generar confusión y conflictos dentro del equipo, lo que podría afectar negativamente la calidad de las decisiones y la cohesión del equipo. Además, en entornos donde existen jerarquías organizacionales rígidas o culturas de trabajo tradicionales, la transición hacia equipos auto organizados puede encontrar resistencia y requerir un cambio cultural significativo para ser efectiva. En resumen, si bien los equipos auto organizados tienen el potencial de impulsar la excelencia en arquitecturas, requisitos y diseños, su éxito depende en gran medida de la madurez y la capacidad de adaptación del equipo, así como del contexto organizacional en el que operan.