Universidad Autónoma de Querétaro

Actividad 2. Estándares

Materia: Pruebas de aseguramiento de la calidad del software

Profesor: Fernando Navarro Villeda

Nombre del alumno: José Ángel Salinas Terrazas

Expediente: 315437

Fecha: 12 de November de 2025

Índice

1 Introducción

Este documento presenta una revisión de los principales estándares y modelos de calidad aplicables al desarrollo de software, específicamente el modelo CMMI (Capability Maturity Model Integration) y la familia de normas ISO 9000. Ambos marcos proporcionan directrices y mejores prácticas para mejorar la calidad, eficiencia y madurez de los procesos de desarrollo de software.

2 Modelo CMMI (Capability Maturity Model Integration)

El modelo CMMI es un marco de mejora de procesos que ayuda a las organizaciones a optimizar sus capacidades en desarrollo de software y sistemas. A continuación se describen las versiones principales del modelo.

2.1 CMMI para Desarrollo v1.3 (2010)

Es un modelo de mejora de procesos enfocado en el desarrollo de software y sistemas. CMMI-DEV v1.3 define 22 áreas de proceso que abarcan prácticas clave de ingeniería de software, gestión de proyectos, aseguramiento de calidad y gestión organizacional. Estas áreas incluyen, por ejemplo, Gestión de Requisitos (REQM), Planificación de Proyectos (PP), Gestión de la Configuración (CM), Solución Técnica (TS), Verificación (VER) y Validación (VAL), entre otras.

El modelo organiza las áreas en niveles de madurez crecientes del 2 al 5, proporcionando una senda de mejora continua de procesos. CMMI-DEV v1.3 fue ampliamente adoptado en la industria de software para evaluar y mejorar la capacidad de los procesos de desarrollo.

Estado: Esta versión (publicada en 2010) fue la referencia principal durante varios años, pero ha sido reemplazada por versiones posteriores (CMMI V2.0 y V3.0).

2.2 CMMI Development V2.0 (2018)

Es la segunda generación del modelo, que unificó en un solo marco las tres áreas originales de CMMI (Desarrollo, Servicios y Adquisición). CMMI V2.0 actualizó las mejores prácticas para el desarrollo de productos y software, introduciendo el concepto de áreas de práctica (antes llamadas áreas de proceso) y poniendo mayor énfasis en la mejora del desempeño y la integración con metodologías ágiles.

Este modelo cubre las mismas disciplinas que CMMI-DEV (gestión de requisitos, diseño técnico, verificación, validación, gestión de proyectos, aseguramiento de calidad, etc.) pero de forma más ágil y flexible, incorporando también prácticas para gestión de riesgos y trabajo con proveedores. CMMI V2.0 mantuvo la estructura de cinco niveles de madurez para evaluar la capacidad organizativa. Incluyó nuevas áreas como Gobernanza e Infraestructura de Implementación para apoyar la institucionalización de los procesos.

Estado: Fue la versión vigente desde 2018 y ampliamente utilizada en evaluaciones hasta inicios de 2023; posteriormente, con la publicación de la versión 3.0 en 2023, CMMI V2.0 ha sido sucedida por la nueva versión (quedando V2.0 obsoleta para futuras certificaciones).

2.3 CMMI Versión 3.0 (2023)

Es la última versión del modelo CMMI (también denominada CMMI Performance Solutions). Amplía el alcance del modelo más allá de las áreas tradicionales de desarrollo, servicios y adqui-

siciones, incorporando nuevos dominios como seguridad, seguridad funcional (safety), gestión de datos, gestión de personal y trabajo remoto, entre otros.

CMMI 3.0 continúa proporcionando un modelo integrado y personalizable de mejores prácticas para mejorar el desempeño organizacional en proyectos de software y otros ámbitos. Mantiene el enfoque en resultados (outcomes) medibles (calidad, cronograma, costo, satisfacción del cliente) y refuerza principios de mejora continua. Esta versión se alinea con las necesidades actuales de la industria tecnológica, permitiendo a las organizaciones de desarrollo de software evaluar y elevar su madurez en un espectro más amplio de capacidades.

Estado: Versión vigente (publicada en 2023) y utilizada para evaluaciones CMMI actuales; ha reemplazado oficialmente a CMMI V2.0.

3 Normas ISO 9000 aplicables al ciclo de vida del software

La familia ISO 9000 es un conjunto de normas internacionales para sistemas de gestión de la calidad, aplicables a cualquier organización, incluyendo las de desarrollo de software.

3.1 ISO 9000:2015

Sistemas de gestión de la calidad – Fundamentos y vocabulario. Norma que proporciona los principios fundamentales y la terminología básica de los sistemas de gestión de la calidad. No contiene requisitos auditables, pero sirve de referencia conceptual para comprender los términos y enfoques de calidad que se emplean en ISO 9001 y otras normas de la familia.

En desarrollo de software, ISO 9000 ayuda a establecer un lenguaje común sobre calidad (definiciones de producto, procedimiento, mejora continua, etc.) y a entender los fundamentos (por ejemplo, enfoque a procesos, orientación al cliente y ciclo de mejora PDCA) que también aplican a procesos de ingeniería de software.

Estado: Versión vigente (2015), publicada conjuntamente con ISO 9001:2015, reemplazando a la edición 2005 anterior.

3.2 ISO 9001:2015

Sistemas de gestión de la calidad – Requisitos. Es la norma principal de la familia ISO 9000 y la única certificable, que especifica los requisitos para implementar un Sistema de Gestión de la Calidad (SGC). Aplica a cualquier organización, incluyendo empresas de desarrollo de software, para asegurar que sus procesos (desde la ingeniería de requisitos y desarrollo hasta pruebas, entrega y mantenimiento) se ejecuten de forma controlada y orientada a la calidad del producto y la satisfacción del cliente.

La ISO 9001:2015 enfatiza el enfoque a procesos, la gestión de riesgos, el liderazgo de la alta dirección, la mejora continua y el enfoque al cliente, todo lo cual es aplicable al ciclo de vida del software. Muchas organizaciones de software adoptan ISO 9001 para estructurar sus procesos de desarrollo y soporte, garantizando consistencia y calidad en sus productos de software.

Estado: Versión vigente publicada en 2015, que sustituyó a la edición anterior ISO 9001:2008. (Se espera una próxima revisión alrededor de 2026, pero la versión 2015 sigue siendo la actual estándar de referencia).

3.3 ISO 9004:2018

Gestión de la calidad – Calidad de una organización – Directrices para lograr el éxito sostenido. Es una norma complementaria a ISO 9001, que brinda directrices para mejorar la eficacia y eficiencia del sistema de calidad de una organización a largo plazo. Mientras ISO 9001 se centra en requisitos mínimos para asegurar la calidad de productos/servicios, ISO 9004 extiende el enfoque hacia la calidad de la organización en su conjunto, promoviendo la mejora del desempeño, la satisfacción de todas las partes interesadas y el éxito sostenido en entornos complejos y cambiantes.

Para una empresa de desarrollo de software, ISO 9004:2018 ofrece recomendaciones para optimizar la gestión de la calidad más allá de lo básico: por ejemplo, cómo evaluar el contexto de

la organización, fomentar el liderazgo y la cultura de calidad, gestionar relaciones con clientes y proveedores, e innovar en procesos para asegurar viabilidad y competitividad a largo plazo.

Estado: Versión vigente (publicada en 2018), la cual reemplazó a ISO 9004:2009. No es una norma certificable, sino una guía de buenas prácticas voluntarias alineadas con los principios de ISO 9001:2015.

3.4 ISO/IEC/IEEE 90003:2018

Ingeniería de software – Directrices para la aplicación de ISO 9001:2015 al software. Norma especializada que adapta los requisitos genéricos de ISO 9001 al ciclo de vida del software. Proporciona orientación detallada para organizaciones que desarrollan, adquieren, operan o mantienen software, sobre cómo implementar un Sistema de Gestión de Calidad conforme a ISO 9001 en actividades específicas de ingeniería de software (por ejemplo, gestión de requisitos de software, diseño y codificación, pruebas, mantenimiento, configuración del software, etc.).

ISO/IEC 90003 no introduce requisitos adicionales, sino que interpreta cada cláusula de ISO 9001:2015 en términos de prácticas de ingeniería de software, facilitando su aplicación en proyectos de software. Esta guía nació originalmente como ISO 9000-3:1997, enfocada en aplicar ISO 9001:1994 al desarrollo de software, y luego fue emitida como ISO/IEC 90003:2004; la edición vigente 90003:2018 alinea las directrices con ISO 9001:2015.

Estado: Versión vigente (2018) – primera edición bajo la triple designación ISO/IEC/IEEE – que actualizó y sustituyó a la ISO/IEC 90003:2014 para reflejar los cambios de ISO 9001:2015. Sigue siendo la referencia actual para la aplicación de ISO 9001 en organizaciones de desarrollo de software.

4 Tablas Comparativas de Estándares

4.1 Modelo CMMI (Capability Maturity Model Integration)

Estándar	Descripción del Estándar	Ejemplo de Aplicación
CMMI v1.3 (2010) *Obsoleto*	Modelo con 22 áreas de proceso organizadas en 5 niveles de madurez. Cubre gestión de requisitos (REQM), planificación de proyectos (PP), gestión de configuración (CM), solución técnica (TS), verificación (VER) y validación (VAL).	**Caso:** Banco Nacional implementa nivel 3 de madurez. **Acciones:** Estandariza gestión de requisitos con trazabilidad, usa métricas históricas para estimación de proyectos, realiza revisiones técnicas por pares en cada módulo, y ejecuta pruebas de aceptación con usuarios antes de release.
CMMI V2.0 (2018) *Obsoleto*	Unifica Desarrollo, Servicios y Adquisición. Introduce áreas de práctica más flexibles, compatible con metodologías ágiles. Incluye gobernanza e infraestructura de implementación.	**Caso:** E-commerce en crecimiento con Scrum. **Acciones:** Define Definition of Done por sprint, realiza retrospectivas con registro de lecciones aprendidas, gestiona SLA con proveedor de pasarela de pagos, evalúa riesgos de fraude en cada iteración.
CMMI V3.0 (2023) *Vigente*	Versión actual con dominios ampliados: seguridad (security), seguridad funcional (safety), gestión de datos, gestión de personal y trabajo remoto. Enfoque en resultados medibles (calidad, costo, tiempo).	**Caso:** FinTech con equipos distribuidos globalmente. **Acciones:** Ejecuta análisis de vulnerabilidades OWASP en CI/CD, implementa cifrado de datos según GDPR, mantiene onboarding virtual de 30 días para nuevos developers, mide SLA de 99.95% uptime.

Tabla 1: Estándares CMMI aplicables al desarrollo de software

4.2 Familia ISO 9000 para Desarrollo de Software

Estándar	Descripción del Estándar	Ejemplo de Aplicación
ISO 9000:2015 *Vigente*	Fundamentos y vocabulario de SGC. No certificable. Define terminología estándar: no conformidad, acción correctiva, acción preventiva, mejora continua, ciclo PDCA.	**Caso:** Startup tecnológica establece lenguaje común. **Acciones:** En dailys usa «No Conformidad» para bugs P0/P1, «Acción Correctiva» para fixes, «Acción Preventiva» para refactoring técnico, documenta mejora continua en retrospectivas.
ISO 9001:2015 *Vigente (Certificable)*	Requisitos para SGC. Única norma certificable de la familia. Enfoca procesos, gestión de riesgos, liderazgo, mejora continua y satisfacción del cliente.	**Caso:** Software factory busca certificación. **Acciones:** Documenta procedimientos (desarrollo, testing, deployment), mantiene matriz de riesgos actualizada, audita internamente cada trimestre, registra NC y acciones correctivas, mide NPS cliente, controla versiones en Git.
ISO 9004:2018 *Vigente*	Directrices para éxito sostenido. No certificable. Complementa ISO 9001 con enfoque en eficacia organizacional, satisfacción de stakeholders y competitividad a largo plazo.	**Caso:** Consultora busca crecimiento sostenible. **Acciones:** Realiza análisis FODA trimestral, implementa programa de mentoring senior-junior, establece partnerships con AWS y Azure, adopta arquitecturas serverless para optimizar costos.
ISO/IEC/IEEE 90003:2018 *Vigente*	Guía para aplicar ISO 9001:2015 específicamente al ciclo de vida del software. Interpreta cada cláusula de ISO 9001 en contexto de ingeniería de software.	**Caso:** Equipo de apps móviles aplica SGC. **Acciones:** Mantiene backlog priorizado con trazabilidad (requisitos), documenta decisiones arquitectónicas en ADR, logra 85% cobertura en tests unitarios, ejecuta beta testing con 100 usuarios, usa GitFlow para gestión de configuración.

Tabla 2: Estándares ISO 9000 aplicables al desarrollo de software

5 Aplicación de Estándares al Proyecto Actual

A continuación se describe cómo se podrían aplicar los estándares CMMI V3.0 (2023) e ISO/IEC/IEEE 90003:2018 al **Sistema de Gestión de Seguros - Seguros Fianzas VILLALOBOS**, un sistema desarrollado con Electron y arquitectura MVC para gestión de seguros, optimizado para hardware de bajo rendimiento (Intel Celeron N4120, 4GB RAM).

5.1 Aplicación de CMMI V3.0 (2023)

5.1.1 Nivel 2: Gestión de Requisitos (REQM)

Cómo aplicarlo: Crear una matriz de trazabilidad de requisitos que vincule cada funcionalidad del sistema con la solicitud original del cliente.

Ejemplo práctico:

- Documentar el requisito «El cliente necesita gestionar pólizas de seguros» como REQ-001
- Vincular REQ-001 con: Modelo `poliza_model.js`, Controlador `polizas_controller.js`, Vista `polizas_partial.html`, y Tests `test_polizas.js`
- Cuando el cliente solicite un cambio (ej: agregar campo «Beneficiario»), actualizar la matriz para rastrear el impacto en código, pruebas y documentación
- Mantener un registro de cambios que justifique por qué se aceptó o rechazó cada modificación

5.1.2 Nivel 2: Planificación de Proyectos (PP)

Cómo aplicarlo: Establecer un plan de proyecto basado en métricas históricas y estimaciones cuantitativas.

Ejemplo práctico:

- Usar el análisis COCOMO realizado (8,731 LOC) para estimar que cada 100 líneas de código requieren aproximadamente 1.5 horas de desarrollo
- Crear un cronograma por módulos: Semana 1-2 (Autenticación y usuarios), Semana 3-4 (Clientes), Semana 5-6 (Pólizas), Semana 7-8 (Recibos y documentos)
- Definir hitos con criterios de aceptación medibles: «Módulo de Clientes completo cuando: CRUD funcional + validaciones + 5 pruebas automatizadas pasando + documentación actualizada»
- Realizar seguimiento semanal comparando líneas de código planeadas vs. reales

5.1.3 Nivel 3: Verificación (VER) y Validación (VAL)

Cómo aplicarlo: Implementar un proceso formal de revisión y pruebas en cada entrega.

Ejemplo práctico para Verificación:

- Antes de integrar cambios al branch principal, realizar code review con checklist: ¿El código sigue el patrón MVC? ¿Tiene validación de entrada? ¿Las funciones están documentadas? ¿Se actualizó el README?
- Ejecutar suite de pruebas automatizadas: `npm run test:db && npm run test:ui`
- Verificar que el consumo de recursos cumple límites: RAM < 500MB, CPU < 40%

Ejemplo práctico para Validación:

- Cada viernes, realizar sesión de validación con un usuario representante del cliente (personal de Seguros VILLALOBOS)

- Usuario ejecuta casos de uso reales: «Registrar nuevo cliente con RFC duplicado y verificar mensaje de error»
- Documentar hallazgos en formato: Caso de uso ejecutado, Resultado esperado, Resultado obtenido, Estado (Aprobado/Rechazado)

5.1.4 Nivel 5 (CMMI V3.0): Seguridad (Security)

Cómo aplicarlo: Integrar análisis de seguridad en el ciclo de desarrollo.

Ejemplo práctico:

- **Análisis de amenazas**: Antes de desarrollar el módulo de login, identificar amenazas: fuerza bruta, inyección SQL, almacenamiento inseguro de contraseñas
- **Controles implementados**: Para cada amenaza, documentar el control: «Amenaza: Contraseñas en texto plano → Control: bcryptjs con 10 rounds de salt»
- **Revisión de código de seguridad**: Usar checklist OWASP Top 10 al revisar código que maneja datos sensibles
- **Auditoría continua**: Cada acción crítica (login, creación de póliza, modificación de cliente) se registra en tabla de auditoría con timestamp, usuario, acción y datos modificados

5.1.5 Nivel 5 (CMMI V3.0): Gestión de Datos

Cómo aplicarlo: Establecer políticas de gestión del ciclo de vida de datos.

Ejemplo práctico:

- **Clasificación de datos**: Identificar qué datos son sensibles (contraseñas, RFCs) vs. públicos (nombres de aseguradoras)
- **Política de respaldos**: Configurar backup automático de la base de datos SQLite cada 6 horas, manteniendo 7 días de historial
- **Retención de datos**: Definir que los registros de auditoría se conservan 1 año, luego se archivan
- **Cifrado**: Cifrar el archivo `seguros.db` en reposo usando AES-256 antes de distribuir el instalador

5.2 Aplicación de ISO/IEC/IEEE 90003:2018

5.2.1 Cláusula 4: Contexto de la Organización

Cómo aplicarlo: Comprender las necesidades y expectativas del cliente para adaptar el SGC.

Ejemplo práctico:

- **Análisis del contexto**: Documentar que el cliente (Seguros VILLALOBOS) es una empresa pequeña, con personal no técnico, usando hardware limitado, sin conexión a internet estable
- **Partes interesadas**: Identificar stakeholders: Gerente (necesita reportes), Operadores (necesitan rapidez), Soporte técnico (necesitan logs), Cliente final (necesitan disponibilidad)
- **Alcance del SGC**: Definir que el SGC aplica al desarrollo, pruebas, distribución y mantenimiento del sistema de escritorio, pero NO al hardware del cliente ni redes externas

5.2.2 Cláusula 7: Soporte (Recursos y Competencias)

Cómo aplicarlo: Asegurar que el equipo de desarrollo tiene las competencias necesarias.

Ejemplo práctico:

- **Matriz de competencias**: Documentar que el proyecto requiere: JavaScript/Node.js (avanzado), Electron (intermedio), SQL (básico), Testing con Playwright (básico)
- **Capacitación**: Si el equipo no conoce Playwright, planificar 1 semana de capacitación antes de iniciar módulo de testing
- **Recursos técnicos**: Asegurar acceso a: Equipo con specs similares al del cliente para pruebas de rendimiento, Licencias de herramientas (Git, IDE), Documentación técnica (Electron docs, SQL.js docs)

5.2.3 Cláusula 8: Operación (Desarrollo de Software)

Cómo aplicarlo: Implementar controles en cada fase del desarrollo.

Ejemplo práctico - Fase de Diseño:
- Crear documento de arquitectura que describa: Patrón MVC elegido, justificación de SQLite vs. otras BD, decisión de usar sql.js por portabilidad
- Diseñar diagrama Entidad-Relación de la base de datos con relaciones Cliente → Póliza → Recibo
- Revisar y aprobar diseño en reunión con cliente: «¿Esta estructura de datos refleja su proceso de negocio?»

Ejemplo práctico - Fase de Codificación:
- Establecer estándar de código: Usar ESLint, nombrar controladores como `*_controller.js`, funciones async con try-catch
- Implementar revisión por pares: Cada Pull Request requiere aprobación de al menos 1 desarrollador antes de merge
- Control de versiones: Usar Git Flow con branches: `main` (producción), `develop` (integración), `feature/*` (nuevas funcionalidades)

5.2.4 Cláusula 9: Evaluación del Desempeño

Cómo aplicarlo: Medir y monitorear la calidad del software continuamente.

Ejemplo práctico - Métricas de Calidad:
- **Defectos por módulo**: Registrar bugs encontrados en testing por cada módulo (Clientes: 5 bugs, Pólizas: 3 bugs)
- **Cobertura de pruebas**: Objetivo mínimo 70% de cobertura en tests automatizados
- **Tiempo de resolución**: Medir días desde que se reporta un bug hasta que se corrige (objetivo: < 5 días para bugs críticos)
- **Satisfacción del cliente**: Cada 2 semanas, encuesta breve: «Del 1-10, ¿qué tan satisfecho está con el avance del proyecto?»

Ejemplo práctico - Auditoría Interna:
- Cada mes, revisar si se están siguiendo los procesos: ¿Se documentaron los requisitos nuevos? ¿Se hicieron code reviews? ¿Se ejecutaron las pruebas?
- Registrar no conformidades: «NC-001: Se integró módulo de Recibos sin ejecutar pruebas de integración»
- Definir acción correctiva: «AC-001: Establecer hook de Git que bloquee push si fallan tests»

5.2.5 Cláusula 10: Mejora Continua

Cómo aplicarlo: Identificar oportunidades de mejora en procesos y producto.

Ejemplo práctico - Ciclo PDCA:

- **Plan**: Detectar problema: «Los tiempos de búsqueda de clientes son lentos (> 2 segundos con 1000 registros)»
- **Do**: Implementar mejora: Agregar índice en columna `rfc` de tabla clientes
- **Check**: Medir resultado: Búsquedas ahora toman < 300ms
- **Act**: Estandarizar: Agregar índices en todas las columnas usadas en búsquedas (nombre, email)

Ejemplo práctico - Retroalimentación del Cliente:

- Después de la entrega v1.0, cliente reporta: «La aplicación consume mucha RAM en nuestras laptops»
- Acción de mejora: Investigar optimizaciones de Electron, implementar limitación de heap a 512MB, reducir frame rate cuando está minimizada
- Medir: Consumo de RAM bajó de 800MB a 350MB
- Documentar lección aprendida para futuros proyectos con hardware limitado

6 Conclusión

Los estándares y modelos de calidad presentados (CMMI V3.0 e ISO/IEC/IEEE 90003:2018) proporcionan marcos robustos y prácticos para mejorar la calidad del software. CMMI, administrado por el CMMI Institute (parte de ISACA), ofrece un modelo de madurez gradual que permite a las organizaciones evolucionar desde procesos básicos hasta prácticas avanzadas que incluyen seguridad, gestión de datos y trabajo remoto. Por su parte, la familia ISO 9000, publicada por la Organización Internacional de Normalización, establece requisitos y directrices para sistemas de gestión de calidad aplicables a cualquier tipo de organización de desarrollo de software.

La sección de aplicación práctica demostró que estos estándares no son solo teoría académica, sino herramientas concretas que se pueden adaptar a proyectos reales. En el contexto del Sistema de Gestión de Seguros VILLALOBOS, se ilustró cómo implementar:

- **Gestión de requisitos con trazabilidad** (CMMI REQM) mediante matrices que vinculan necesidades del cliente con código y pruebas
- **Planificación basada en métricas** (CMMI PP) usando estimaciones COCOMO y cronogramas por módulos
- **Verificación y validación sistemática** (CMMI VER/VAL) con code reviews, pruebas automatizadas y sesiones de validación con usuarios
- **Seguridad integrada** (CMMI V3.0 Security) mediante análisis de amenazas y controles documentados
- **Gestión del ciclo de vida completo** (ISO 90003) desde análisis de requisitos hasta mantenimiento
- **Mejora continua** (ISO 90003 Cláusula 10) aplicando ciclo PDCA con métricas concretas

Lo más relevante es que estos estándares son escalables: no se requiere una certificación formal ni procesos burocráticos para beneficiarse de sus principios. Incluso en proyectos pequeños o medianos, aplicar prácticas como documentación de requisitos, control de versiones estructurado, revisiones de código, testing automatizado, y métricas de calidad puede marcar la diferencia entre un software frágil y uno robusto, mantenible y alineado con las expectativas del cliente. La clave está en adaptar los estándares al contexto específico del proyecto, enfocándose en aquellas prácticas que generen mayor valor.

Referencias

CMMI Institute. (2018). *CMMI Model Version 2.0* [Technical report].

CMMI Institute. (2023). *CMMI Content Changes* [Technical report]. https://cmmiinstitute.com/getattachment/47a7c84e-472c-4f7f-a473-ddc21c6ae045/attachment.aspx

International Organization for Standardization. (2015a). *Quality management systems — Fundamentals and vocabulary* (No. ISO9000:2015). https://www.iso.org/standard/45481.html

International Organization for Standardization. (2015b). *Quality management systems — Requirements* (No. ISO9001:2015). https://www.iso.org/standard/62085.html

International Organization for Standardization. (2018). *Quality management — Quality of an organization — Guidance to achieve sustained success* (No. ISO9004:2018). https://www.iso.org/standard/70397.html

International Organization for Standardization, International Electrotechnical Commission, & Institute of Electrical and Electronics Engineers. (2018). *Software engineering — Guidelines for the application of ISO 9001:2015 to computer software* (No. ISO/IEC/IEEE90003:2018). https://www.iso.org/standard/74348.html

Software Engineering Institute. (2010). *CMMI for Development, Version 1.3* (technical report No. CMU/SEI-2010-TR-033). https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9661