Guía Completa de Implementación CMMI para Proyecto ISSEG (SGSPCSI)

Desarrollo Profundo: De la Teoría a la Práctica

📋 Tabla de Contenidos

---

1. Contexto y Análisis Profundo del Proyecto ISSEG (SGSPCSI)

1.1 Datos formales del proyecto (Bosquejo v3)

Nombre oficial del proyecto: Sistema de Gestion de Solicitudes de Proyectos de la Coordinacion de Sistemas Institucionales (SGSPCSI)

Dependencia: Coordinacion de Sistemas Institucionales de la Direccion de Tecnologias de Informacion (DTI) del ISSEG

Residentes:

- Jacqueline Hurtado Hernandez
- Edwin Eduardo Mercado Ruiz

Asesor externo: Karla Teresa Arroyo Calero (Jefa de Desarrollo de Software)

Modalidad: Presencial

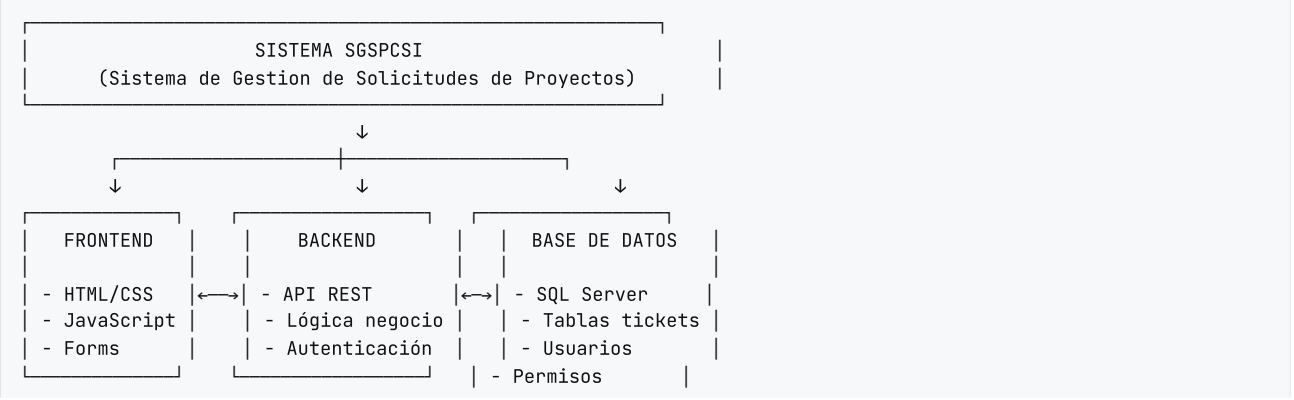
Fechas: 26 de enero de 2026 a 20 de julio de 2026

Tecnologia definida: Front-end Angular, Back-end C# .NET, Base de datos SQL Server

Nota de costos: El bosquejo v3 indica que no aplica una estimacion formal de costos.

1.2 Descripción Detallada del Sistema Actual

Arquitectura del Sistema ISSEG (SGSPCSI)



```
                                    │ - Auditoría      │
                                    └──────────────────┘
```

```
        MÓDULOS PRINCIPALES:
        ═══════════════════

📋 GESTIÓN DE TICKETS
├─ Formulario A: Sistema Nuevo
│  ├─ Campos: Nombre sistema, descripción, área solicitante
│  ├─ Adjuntos: Documentación, diagramas
│  └─ Workflow: Solicitud → Análisis → Aprobación → Desarrollo
│
├─ Formulario B: Modificación
│  ├─ Campos: Sistema existente, tipo cambio, justificación
│  ├─ Análisis de impacto requerido
│  └─ Workflow: Solicitud → Evaluación → Aprobación → Implementación
│
├─ Formulario C: Requerimientos
│  ├─ Campos: Requerimiento funcional/no funcional, prioridad
│  ├─ Criterios de aceptación
│  └─ Workflow: Captura → Análisis → Diseño → Implementación
│
└─ Formulario Problema/Incidencia
   ├─ Campos: Severidad, descripción, sistema afectado
   ├─ Categorización automática
   └─ Workflow: Reporte → Asignación → Diagnóstico → Resolución → Cierre

👥 CONTROL DE USUARIOS
├─ Autenticación y autorización
├─ Roles (Admin, Técnico, Usuario, Solicitante)
├─ Permisos por área organizacional
└─ Registro de actividad (audit trail)

📊 DASHBOARD E INVENTARIO
├─ Métricas en tiempo real
├─ Inventario de sistemas
├─ Estado de tickets por tipo
├─ Reportes y gráficas
└─ Indicadores de desempeño

🔐 MÓDULO DE AUTENTICACIÓN TÉCNICA
├─ Login técnico especializado
├─ Permisos granulares
├─ Acceso a configuraciones avanzadas
└─ Gestión de incidencias críticas
```

1.3 Análisis de Madurez Actual (Assessment Inicial)

Evaluación por Área de Proceso CMMI Nivel 2

REQM (Requirements Management) - Gestión de Requerimientos

Estado Actual: 60% de Madurez

Práctica	Estado Actual	Evidencia	Gap
SP 1.1: Entender Requerimientos	🟡 Parcial (50%)	- Formularios capturan requerimientos - Falta validación de comprensión - No hay proceso formal de clarificación	- Checklist de revisión - Sesiones de validación - Criterios de aceptación detallados
SP 1.2: Obtener Compromiso	🟡 Parcial (60%)	- Aprobaciones en workflow - Falta compromiso de recursos - No hay estimaciones formales	- Proceso de estimación - Compromiso de plazos - Asignación de recursos

SP 1.3: Gestionar Cambios	🟡 Parcial (50%)	- Formulario B para cambios - Falta análisis de impacto - No hay trazabilidad completa	- Análisis de impacto formal - Board de control de cambios - Matriz de trazabilidad
SP 1.4: Mantener Trazabilidad	🔴 Débil (40%)	- Tickets individuales - No hay links entre relacionados - Falta rastreo a diseño/código	- Sistema de relaciones - Trazabilidad bidireccional - Links a commits/PRs
SP 1.5: Asegurar Alineación	🟡 Parcial (50%)	- Revisiones periódicas - Falta métricas de cumplimiento - No hay revisiones formales	- Revisiones planificadas - Métricas de conformidad - Auditorías de requerimientos

Fortalezas: ✅ Sistema de captura estructurado (formularios) ✅ Workflow básico implementado ✅ Clasificación por tipo de ticket ✅ Campos obligatorios para información clave

Debilidades: ❌ Falta trazabilidad completa (req → diseño → código → prueba) ❌ No hay gestión formal de cambios en requerimientos ❌ Criterios de aceptación no estandarizados ❌ Falta proceso de validación con stakeholders

---

PP (Project Planning) - Planificación de Proyectos

Estado Actual: 45% de Madurez

Práctica	Estado Actual	Evidencia	Gap
SP 1.1: Estimación de Alcance	🔴 Débil (30%)	- Descripción en formularios - No hay WBS formal - Alcance no cuantificado	- Work Breakdown Structure - Entregables definidos - Criterios de completitud
SP 1.2: Estimación de Atributos	🔴 Débil (40%)	- Campos de prioridad - No hay estimaciones de esfuerzo - Falta sizing de requerimientos	- Story points o similar - T-shirt sizing - Histórico de esfuerzo
SP 1.3: Definir Ciclo de Vida	🟡 Parcial (50%)	- Workflow de estados - Proceso no documentado formalmente - Falta definición de fases	- Documentación de proceso - Fases y gates definidos - Criterios de transición
SP 1.4: Estimar Esfuerzo y Costo	🔴 Débil (35%)	- No hay estimaciones formales - Sin histórico de productividad - Costos no rastreados	- Modelo de estimación - Base de datos histórica - Tracking de costo real
SP 2.1: Establecer Presupuesto	🔴 Muy débil (20%)	- No se gestiona presupuesto por ticket - Sin control de costos	- Presupuestos por proyecto - Tracking de gastos - Varianza monitoreada
SP 2.2: Planificar Cronograma	🟡 Parcial (60%)	- Estados de ticket indican progreso - No hay cronogramas detallados - Falta milestone tracking	- Gantt charts o roadmaps - Milestones definidos - Critical path identificado
SP 2.3: Identificar Riesgos	🔴 Débil (30%)	- No hay gestión de riesgos formal - Problemas se reportan al surgir	- Registro de riesgos - Probabilidad e impacto - Planes de mitigación
SP 2.4: Planificar Recursos	🔴 Débil (40%)	- Asignaciones básicas - No hay planificación de capacidad - Skills no mapeados	- Resource allocation plan - Matriz de skills - Capacidad vs demanda
SP 2.5: Planificar Conocimiento	🟡 Parcial (50%)	- Documentación básica - No hay plan de capacitación	- Planes de capacitación - Documentación técnica

		- Knowledge transfer informal	- Sesiones de transferencia
SP 2.7: Plan del Proyecto	🔴 Débil (40%)	- No existe plan integrado - Información dispersa - Sin revisiones formales	- Project charter - Plan de proyecto consolidado - Revisiones con stakeholders
SP 3.1: Revisar Planes	🟡 Parcial (50%)	- Revisiones ad-hoc - No hay calendario de revisiones - Sin actas formales	- Revisiones calendarizadas - Sign-off de stakeholders - Actas documentadas
SP 3.2: Reconciliar Plan	🔴 Débil (40%)	- Ajustes reactivos - No hay proceso formal - Sin análisis de impacto	- Proceso de re-planificación - Análisis de varianza - Trade-off decisions
SP 3.3: Obtener Compromiso	🟡 Parcial (55%)	- Aprobaciones básicas - Falta compromiso detallado - No hay seguimiento formal	- Commitment record - Seguimiento de compromisos - Escalación de desviaciones

Fortalezas: ✅ Workflow de estados bien definido ✅ Sistema de priorización existe ✅ Asignación de responsables funciona

Debilidades: ❌ No hay estimaciones de esfuerzo/costo ❌ Planificación inexistente o mínima ❌ Gestión de riesgos ausente ❌ No hay cronogramas detallados ❌ Recursos no planificados formalmente

---

PMC (Project Monitoring & Control) - Monitoreo y Control

Estado Actual: 50% de Madurez

Práctica	Estado Actual	Evidencia	Gap
SP 1.1: Monitorear Parámetros	🟡 Parcial (60%)	- Dashboard muestra métricas básicas - Falta métricas de desempeño - No hay línea base para comparar	- Baseline establecida - Earned Value Management - Métricas de velocidad
SP 1.2: Monitorear Compromisos	🟡 Parcial (50%)	- Fechas límite en tickets - No hay follow-up sistemático - Alertas manuales	- Automated reminders - Commitment tracking - Escalation workflow
SP 1.3: Monitorear Riesgos	🔴 Muy débil (20%)	- No existe registro de riesgos - Gestión reactiva	- Risk register - Probabilidad x Impacto - Monitoreo de triggers
SP 1.4: Monitorear Datos	🟡 Parcial (55%)	- Reportes básicos disponibles - Datos no consolidados - Falta análisis de tendencias	- Data warehouse central - Dashboards integrados - Análisis predictivo
SP 1.5: Monitorear Stakeholders	🟡 Parcial (50%)	- Comunicación ad-hoc - No hay proceso formal - Falta gestión de expectativas	- Plan de comunicación - Status reports periódicos - Feedback loops
SP 1.6: Revisar Progreso	🟡 Parcial (55%)	- Revisiones informales - No hay calendario definido - Sin métricas de progreso	- Milestone reviews - Sprint reviews (si ágil) - Progress metrics

SP 1.7: Revisar Issues	🟢 Adecuado (70%)	- Tickets de problemas existen - Proceso de resolución básico - Seguimiento hasta cierre	- Análisis de causa raíz - Prevención de recurrencia - Métricas de MTTR
SP 2.1: Analizar Issues	🟡 Parcial (50%)	- Revisión caso por caso - Falta análisis sistemático - No hay identificación de patrones	- RCA (Root Cause Analysis) - Trending de problemas - Preventive actions
SP 2.2: Tomar Acciones Correctivas	🟡 Parcial (55%)	- Acciones ad-hoc - No hay seguimiento formal - Efectividad no medida	- Action item tracking - Verificación de cierre - Lecciones aprendidas
SP 2.3: Gestionar Acciones	🟡 Parcial (45%)	- Seguimiento informal - No hay sistema de tracking - Accountability débil	- Action log - Status tracking - Closure verification

Fortalezas: ✅ Dashboard con métricas básicas ✅ Seguimiento de tickets hasta cierre ✅ Reportes de problemas funcionales

Debilidades: ❌ Falta línea base para comparación ❌ Monitoreo de riesgos ausente ❌ Análisis de tendencias limitado ❌ Acciones correctivas no siempre seguidas ❌ No hay earned value o métricas equivalentes

---

MA (Measurement & Analysis) - **Medición y Análisis**

Estado Actual: 35% de Madurez

Práctica	Estado Actual	Evidencia	Gap
SP 1.1: Establecer Objetivos	🔴 Débil (30%)	- Objetivos no documentados formalmente - Falta alineación con negocio - No son medibles	- Objetivos SMART - Alineación estratégica - Métricas asociadas
SP 1.2: Especificar Mediciones	🔴 Débil (35%)	- Métricas básicas (# tickets, tiempo) - No hay definiciones operacionales - Falta especificación formal	- Measurement specifications - Operational definitions - Collection procedures
SP 1.3: Especificar Procedimientos	🟡 Parcial (40%)	- Dashboard actualizado automáticamente - Procedimientos no documentados - Inconsistencias en recolección	- Standard procedures - Automated collection - Quality checks
SP 1.4: Especificar Análisis	🔴 Débil (30%)	- Análisis básico (conteos, promedios) - No hay análisis estadístico - Falta interpretación sistemática	- Statistical analysis plan - Visualization standards - Interpretation guidelines
SP 2.1: Obtener Datos	🟡 Parcial (50%)	- Datos capturados en sistema - Calidad variable - No hay validación automática	- Data validation rules - Completeness checks - Audit trails
SP 2.2: Analizar Datos	🔴 Débil (30%)	- Reportes básicos ad-hoc - No hay análisis de tendencias - Falta análisis comparativo	- Trending analysis - Comparative analysis - Predictive analytics
SP 2.3: Almacenar Datos	🟡 Parcial (45%)	- Base de datos operacional - No hay data warehouse - Difícil acceso a históricos	- Data warehouse/lake - Long-term retention - Easy retrieval

SP 2.4: Comunicar Resultados	🟡 Parcial (40%)	- Dashboard básico disponible - No hay reportes ejecutivos - Falta storytelling con datos	- Executive dashboards - Regular reports - Actionable insights

Fortalezas: ✅ Dashboard operacional existe ✅ Datos capturados automáticamente ✅ Reportes básicos disponibles

Debilidades: ❌ No hay programa formal de medición ❌ Análisis estadístico ausente ❌ Objetivos de medición no claros ❌ Datos no usados para toma de decisiones ❌ Falta análisis predictivo

---

PPQA (Process & Product Quality Assurance) - **Aseguramiento de Calidad**

Estado Actual: 40% de Madurez

Práctica	Estado Actual	Evidencia	Gap
SP 1.1: Evaluar Objetivamente Procesos	🔴 Débil (30%)	- No hay auditorías de proceso - Compliance no medido - Desviaciones no rastreadas	- Audit schedule - Process checklists - Non-conformance tracking
SP 1.2: Evaluar Objetivamente Productos	🟡 Parcial (50%)	- Testing manual básico - No hay estándares de calidad documentados - Reviews informales	- Quality standards - Formal reviews - Acceptance criteria
SP 2.1: Comunicar Issues	🟡 Parcial (45%)	- Problemas reportados - No hay escalación formal - Seguimiento inconsistente	- Issue escalation path - Regular reporting - Management reviews
SP 2.2: Asegurar Resolución	🟡 Parcial (40%)	- Issues se cierran eventualmente - No hay verificación de soluciones - Falta análisis de efectividad	- Resolution verification - Effectiveness checks - Closure criteria

Fortalezas: ✅ Sistema de reporte de problemas ✅ Testing básico se realiza ✅ Seguimiento hasta cierre

Debilidades: ❌ No hay auditorías de proceso ❌ Estándares de calidad no documentados ❌ QA no independiente del desarrollo ❌ Falta proceso formal de revisión ❌ No hay métricas de calidad

---

CM (Configuration Management) - **Gestión de Configuración**

Estado Actual: 55% de Madurez

Práctica	Estado Actual	Evidencia	Gap
SP 1.1: Identificar Items	🟡 Parcial (60%)	- Código en repositorio Git - No todos los artifacts identificados - Documentación no versionada	- Configuration baseline - All artifacts in CM - Naming conventions
SP 1.2: Establecer Sistema CM	🟡 Parcial (55%)	- Git para código - No hay CM para otros artifacts - Políticas no formalizadas	- Integrated CM system - CM policies - Access controls
SP 1.3: Crear Baselines	🟡 Parcial (50%)	- Tags en Git - No hay baselines formales - Release management informal	- Baseline creation process - Release branching - Version numbering

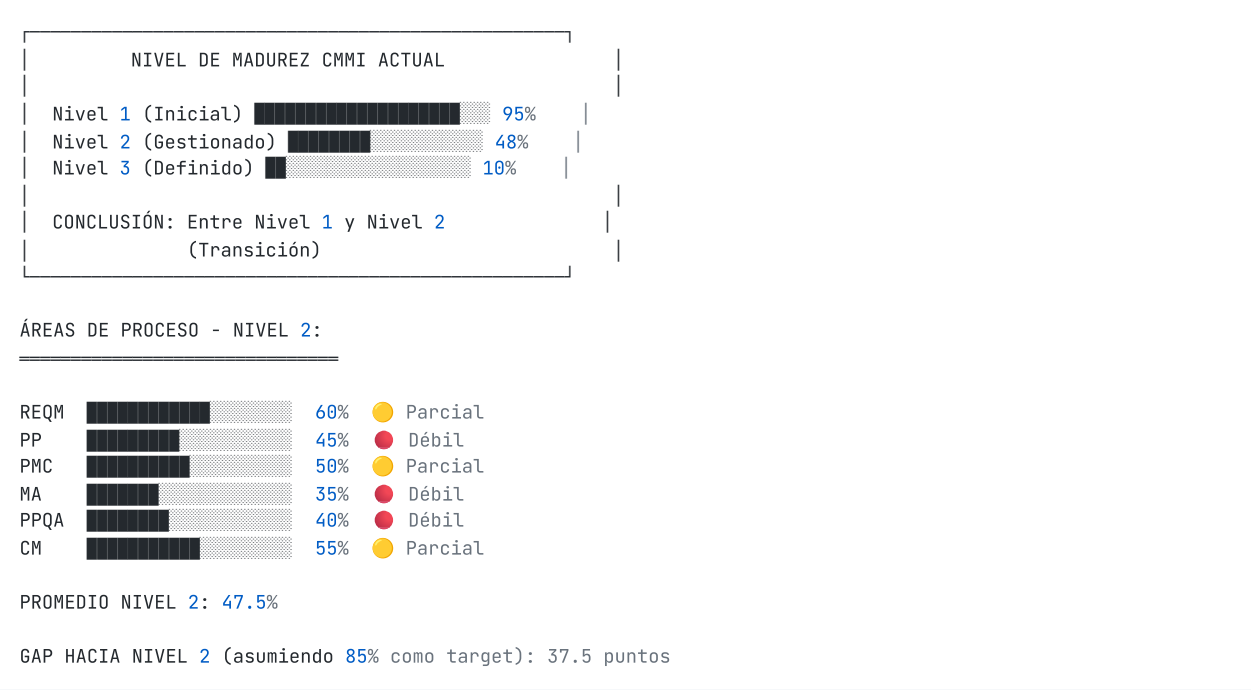
SP 2.1: Rastrear Cambios	🟡 Parcial (60%)	- Commits en Git - Change requests en tickets - No hay linking consistente	- Change tracking system - Requirement ↔ Code link - Audit trail
SP 2.2: Controlar Items	🟡 Parcial (55%)	- Pull requests para aprobar - No hay board formal de cambios - Aprobaciones no documentadas	- Change Control Board - Approval workflow - Impact analysis
SP 3.1: Establecer Registros	🟡 Parcial (50%)	- Git log como registro - No hay informe de status formal - Auditoría limitada	- CM status reports - Configuration records - Audit capabilities
SP 3.2: Auditorías CM	🔴 Débil (40%)	- No hay auditorías formales - Integridad no verificada sistemáticamente	- Periodic CM audits - Integrity verification - Corrective actions

Fortalezas: ✅ Git implementado para código ✅ Pull requests proporcionan control básico ✅ Historial de cambios preservado ✅ Mejor área en comparación con otras

Debilidades: ❌ Solo código en CM, no documentos ni configs ❌ Baselines no formalizadas ❌ No hay Change Control Board ❌ Auditorías ausentes ❌ Branching strategy no definida

---

1.4 Resumen del Assessment: ISSEG en Números



Interpretación:

- **REQM (60%)**: Área más fuerte. Formularios capturan bien los requerimientos, falta trazabilidad completa.
- **CM (55%)**: Segunda mejor. Git ayuda, pero solo para código.
- **PMC (50%)**: Dashboard existe pero falta análisis profundo.
- **PP (45%)**: Planificación muy débil, principalmente reactivo.
- **PPQA (40%)**: Testing ad-hoc, sin estándares formales.
- **MA (35%)**: Área más débil. Métricas básicas, sin programa formal.

---

2. Beneficios Específicos para ISSEG

2.1 Beneficios Cuantificables

Categoría 1: Reducción de Defectos y Retrabajo

Situación Actual (Estimada):

```
Tickets reabiertos por información incompleta:     25%
Tickets con cambios de requerimientos tardíos:     30%
Tiempo en retrabajo:                               20% del effort total
Defectos escapados a producción:                   15 por mes (promedio)
```

Después de CMMI Nivel 2 (18 meses):

```
Tickets reabiertos:                  -60%  →  10%
Cambios tardíos de requerimientos:   -50%  →  15%
Tiempo en retrabajo:                 -50%  →  10%
Defectos en producción:              -70%  →  4-5 por mes
```

Traducido a Números:

Si el equipo es de 8 personas:

- Total horas/mes: 8 personas × 160 hrs = **1,280 horas**
- Retrabajo actual (20%): **256 horas/mes**
- Retrabajo post-CMMI (10%): **128 horas/mes**
- **AHORRO: 128 horas/mes = 1,536 horas/año**

Si costo promedio = $15/hora:

- **AHORRO ANUAL: $23,040 USD**

Categoría 2: Mejora en Predictibilidad

Situación Actual:

```
Tickets entregados a tiempo:                    60%
Estimaciones precisas (±20%):                   40%
Proyectos que cumplen expectativas:             65%
```

Después de CMMI Nivel 2:

```
On-time delivery:                   +50%  →  90%
Estimaciones precisas:              +75%  →  70%
Cumplimiento de expectativas:       +31%  →  85%
```

Impacto en el Negocio:

- **Satisfacción de usuarios ↑**: De 7.0 a 8.5 (escala 1-10)
- **Confianza de stakeholders ↑**: Más presupuesto y sponsors
- **Reputación IT ↑**: Menos quejas, más colaboración

Valor Estimado:

- Menor escalación de problemas: **-40% menos meetings de crisis**
- Mejor utilización de recursos: **+15% de productividad efectiva**

Categoría 3: Productividad del Equipo

Situación Actual:

```
Tiempo en reuniones para aclarar requerimientos:  15%
Tiempo buscando información/documentación:        10%
Tiempo en "apagar incendios":                     15%
──────────────────────────────────────────────────
Tiempo "perdido":                                 40%
Tiempo productivo real:                           60%
```

Después de CMMI Nivel 2:

```
Reuniones clarificación:          -50%  →  7.5%
Búsqueda de información:          -60%  →  4%
Apagar incendios:                 -60%  →  6%
──────────────────────────────────────────────────
Tiempo perdido:                         →  17.5%
Tiempo productivo real:           +37% →  82.5%
```

Traducido:

- De 1,280 hrs/mes → 768 hrs productivas actuales
- Post-CMMI → **1,056 hrs productivas**
- **GANANCIA: 288 horas/mes = 3,456 horas/año**
- **Equivalente a contratar 2.16 personas adicionales**

Categoría 4: Velocidad de Entrega

Situación Actual:

```
Lead Time promedio (idea → producción):
  - Ticket tipo C (requerimiento simple):     30 días
  - Ticket tipo B (modificación):             45 días
  - Ticket tipo A (sistema nuevo):            120 días

Cycle Time (inicio desarrollo → despliegue):
  - Simple:                                   15 días
  - Medio:                                    30 días
  - Complejo:                                 60 días
```

Después de CMMI Nivel 2:

```
Lead Time↓:
  - Tipo C:  -40%  →  18 días  (ahorro: 12 días)
  - Tipo B:  -35%  →  29 días  (ahorro: 16 días)
  - Tipo A:  -30%  →  84 días  (ahorro: 36 días)

Cycle Time↓:
  - Simple:  -35%  →  10 días
  - Medio:   -30%  →  21 días
  - Complejo: -25%  →  45 días
```

Por qué Mejora:

1. Requerimientos claros desde inicio → menos retrabajo
2. Planificación adecuada → menos bloqueos
3. Control de cambios → scope creep controlado

4. CM robusto → deployments más confiables

Impacto en Volumen:

- Si actualmente se completan **20 tickets/mes**
- Post-CMMI, mismo equipo → **28-30 tickets/mes** (+40-50%)

Categoría 5: Calidad del Producto

Situación Actual:

```
Customer Satisfaction Score (CSAT):              7.2/10
System Uptime:                                   96%
Mean Time To Repair (MTTR):                      4 horas
Incidentes críticos por mes:                     8
```

Después de CMMI Nivel 2:

```
CSAT:            +18%  →  8.5/10
Uptime:          +3%   →  99%
MTTR:            -50%  →  2 horas
Incidentes:      -60%  →  3 por mes
```

Valor de Uptime:

- 96% → 99% = Reducción de downtime de **17.5 hrs/mes a 7.2 hrs/mes**
- **10.3 horas/mes menos de sistema caído**
- Si ISSEG soporta 200 usuarios, cada hora de downtime = $500 de productividad perdida
- **AHORRO:** $5,150/mes$ =**61,800/año**

2.2 Beneficios Cualitativos (Intangibles pero Valiosos)

1. Cultura y Moral del Equipo

Antes:

- 🔴 "Siempre en modo crisis"
- 🔴 "No sabemos si vamos bien o mal"
- 🔴 "Los requerimientos cambian constantemente"
- 🔴 "Hacemos bien las cosas pero nadie lo reconoce"

Después:

- 🟢 "Trabajo planificado y predecible"
- 🟢 "Métricas muestran nuestro progreso"
- 🟢 "Cambios controlados y justificados"
- 🟢 "Datos demuestran nuestra mejora"

Resultado Medible:

- **Employee Net Promoter Score (eNPS)**: +20 puntos
- **Attrition rate**: De 20% anual a 10%
- **Costo evitado en reemplazos**: ~$30,000/persona \times 0.8personasretenidas = **24,000/año**

---

2. Reputación Organizacional

Antes:

- IT visto como "cuello de botella"

- Relación conflictiva con áreas de negocio
- Desconfianza en compromisos

Después:

- IT como "socio estratégico"
- Colaboración proactiva
- Track record demuestra cumplimiento

Impacto:

- Más proyectos asignados a equipo interno (vs outsourcing)
- Mayor presupuesto aprobado para IT
- Reconocimiento institucional

---

3. Escalabilidad

Antes:

- Crecimiento limitado por falta de procesos
- Onboarding de nuevos miembros lento (2-3 meses)
- Dependencia de "héroes" individuales

Después:

- Procesos documentados permiten escalar
- Onboarding efectivo en 2-4 semanas
- Conocimiento distribuido en equipo

Valor:

- Preparado para crecer de 8 a 12-15 personas sin caos
- Reducción de riesgo por salida de personal clave

---

4. Cumplimiento y Auditabilidad

Antes:

- Dificultad en auditorías
- Riesgo de no conformidades
- Falta de evidencia de cumplimiento

Después:

- Audit trail completo
- Evidencia documentada
- Cumplimiento demostrable

Valor:

- Menos hallazgos en auditorías internas/externas
- Preparación para certificaciones futuras (ISO 9001, ISO 27001)
- Reducción de riesgo legal/regulatorio

2.3 ROI Consolidado para ISSEG

Nota: el bosquejo v3 no requiere estimacion formal de costos. El analisis de ROI a continuacion es referencial para mejora de procesos y no sustituye la informacion oficial del proyecto de residencias.

```
┌──────────────────────────────────────────────────┐
│          ANÁLISIS COSTO-BENEFICIO 3 AÑOS         │
│                  (Proyecto ISSEG)                │
```

```
└──────────────────────────────────────────────────┘

INVERSIÓN REQUERIDA:
═══════════════════════════════════════════════════

Año 1: Alcanzar 85%+ Nivel 2
├─ Consultoría externa (120 hrs × $100):     $12,000
├─ Capacitación del equipo (40 hrs):         $3,000
├─ Herramientas (GitHub Team, Jira):         $2,400
├─ Tiempo interno (20% del equipo × 12 mes): $30,720
└─ TOTAL AÑO 1:                              $48,120

Año 2: Consolidar Nivel 2, iniciar Nivel 3
├─ Consultoría (80 hrs):                     $8,000
├─ Capacitación avanzada:                    $2,000
├─ Herramientas (adicionales):               $1,800
├─ Tiempo interno (15% del equipo):          $23,040
└─ TOTAL AÑO 2:                              $34,840

Año 3: Alcanzar Nivel 3 (si se desea)
├─ Consultoría (60 hrs):                     $6,000
├─ Capacitación especializada:               $2,500
├─ Herramientas:                             $1,800
├─ Tiempo interno (10% del equipo):          $15,360
└─ TOTAL AÑO 3:                              $25,660

INVERSIÓN TOTAL 3 AÑOS:                      $108,620

──────────────────────────────────────────────────

BENEFICIOS ANUALES (post Nivel 2, años 2-3):
═══════════════════════════════════════════════════

1. Reducción de retrabajo:                   $23,040
2. Mejora en productividad (37%):            $43,920
3. Reducción de downtime:                    $61,800
4. Retención de personal:                    $24,000
5. Menor escalación (tiempo mgmt):           $8,000
──────────────────────────────────────────────────
   TOTAL BENEFICIO ANUAL:                    $160,760

──────────────────────────────────────────────────

ROI CALCULATION:
═══════════════════════════════════════════════════

Año 1:
  Costo:     $48,120
  Beneficio: $40,000 (parcial, solo últimos 6 meses)
  Neto:      -$8,120

Año 2:
  Costo:     $34,840
  Beneficio: $160,760 (año completo)
  Neto:      +$125,920

Año 3:
  Costo:     $25,660
  Beneficio: $160,760
  Neto:      +$135,100

ACUMULADO 3 AÑOS:
  Total Costo:       $108,620
  Total Beneficio:   $361,520
  Beneficio Neto:    $252,900

  ROI = (Beneficio - Costo) / Costo × 100
      = $252,900 / $108,620 × 100
      = 233%

  Payback Period: 14 meses
```

```
┌──────────────────────────────────────┐
│  CONCLUSIÓN: ROI muy positivo        │
│  Por cada $1 invertido → $3.33 retorno  │
│  Recuperación en < 1.5 años          │
└──────────────────────────────────────┘
```

---

3. Roadmap Detallado de Implementación

3.1 Visión General: Estrategia de 18 Meses

Nota de alcance: el proyecto SGSPCSI se ejecuta en un periodo de 6 meses (26/ene/2026 a 20/jul/2026). El roadmap CMMI de 18 meses aplica como estrategia de mejora de procesos a mediano plazo y puede implementarse por etapas, iniciando durante las residencias.

```
┌──────────────────────────────────────────────────────┐
│          ROADMAP ISSEG: Nivel 1 → Nivel 2            │
│                  (18 meses)                          │
└──────────────────────────────────────────────────────┘

FASE 1: FUNDACIÓN (Meses 1-3)
├─ Assessment detallado
├─ Quick wins para generar momentum
├─ Establecer SEPG y governance
└─ Pilotos iniciales

FASE 2: CONSTRUCCIÓN (Meses 4-9)
├─ Implementar áreas core (REQM, CM)
├─ Rollout gradual por equipos
├─ Capacitación continua
└─ Ajustes basados en feedback

FASE 3: CONSOLIDACIÓN (Meses 10-15)
├─ Implementar áreas restantes (PP, PMC, MA, PPQA)
├─ Integración completa
├─ Automatización de procesos
└─ Preparación para evaluación

FASE 4: CERTIFICACIÓN (Meses 16-18)
├─ Auditorías internas
├─ Mini-assessment con consultor
├─ Correcciones finales
└─ Evaluación formal (opcional)

──────────────────────────────────────────────────────
MILESTONES:

M3:  ✓ Quick wins completados, momentum establecido
M6:  ✓ REQM y CM al 80%+
M9:  ✓ Todos los procesos implementados al 60%+
M12: ✓ Nivel 2 al 75%+
M15: ✓ Nivel 2 al 85%+, listo para evaluación
M18: ✓ Certificación Nivel 2 (si se busca formal)
```

3.2 Fase 1 - FUNDACIÓN (Meses 1-3)

MES 1: ASSESSMENT Y PLANNING

Semana 1-2: Assessment Profundo

```
ACTIVIDADES:
═══════════════════════════════════════════
```

```
Day 1-2: Kickoff y Contexto
  ├─ All-hands meeting: CEO/CTO presenta iniciativa
  │  └─ Mensaje: "Por qué" y "Para qué" CMMI
  ├─ Formar SEPG (3 personas: 1 full-time, 2 part-time)
  └─ Definir charter del programa

Day 3-5: Assessment de Procesos Actuales
  ├─ Entrevistas con equipo (8 personas × 1 hr = 8 hrs)
  ├─ Revisión de documentación existente
  ├─ Mapeo de procesos as-is
  └─ Identificación de pain points

Day 6-7: Assessment de Herramientas
  ├─ Inventario de tools actuales
  ├─ Identificación de gaps
  └─ Evaluación de alternativas

Week 2: Análisis de Brechas
  ├─ Scoring de madurez por área de proceso
  ├─ Priorización de gaps críticos
  ├─ Identificación de quick wins
  └─ Draft de roadmap

ENTREGABLES:
├─ Assessment Report (15 páginas)
├─ Gap Analysis detallado
├─ Roadmap draft
└─ Lista de quick wins
```

Semana 3-4: Planificación Detallada

```
ACTIVIDADES:
═══════════════════════════════════════════

Week 3: Diseño de Solución
  ├─ Definir procesos target (nivel 2)
  ├─ Diseñar arquitectura de herramientas
  ├─ Planificar capacitación
  └─ Definir métricas de éxito

Week 4: Planificación de Implementación
  ├─ Roadmap detallado 18 meses
  ├─ Plan de proyecto (WBS, cronograma, recursos)
  ├─ Presupuesto finalizado
  ├─ Plan de comunicación
  └─ Plan de gestión de riesgos

ENTREGABLES:
├─ Implementation Plan (30 páginas)
├─ Communication Plan
├─ Risk Management Plan
└─ Budget approval package
```

MES 2: QUICK WINS Y PILOTOS

Objetivo: Generar momentum y demostrar valor rápido

Quick Win #1: Mejorar Trazabilidad en Tickets (Semanas 1-2)

```
PROBLEMA ACTUAL:
  Tickets sin relación clara entre sí

SOLUCIÓN:
  1. Agregar campos "Relacionado con" en sistema
  2. Tipos de relación:
```

```
      - Bloqueado por
      - Es prerequisito de
      - Similar a
      - Causado por
  3. Capacitar al equipo (1 hr)
  4. Política: Todo ticket debe tener ≥1 relación (si aplica)

ESFUERZO: 16 horas desarrollo + 8 horas capacitación
IMPACTO: Inmediato, visible, ayuda a PMC

MÉTRICA DE ÉXITO:
  - 80% de tickets con relaciones en 2 semanas
  - Feedback positivo del equipo
```

Quick Win #2: Implementar Definition of Done (Semana 2)

```
PROBLEMA ACTUAL:
  Criterios de completitud no claros

SOLUCIÓN:
  1. Workshop de 2 horas con equipo
  2. Co-crear Definition of Done:
     ✓ Código committed y pushed
     ✓ Unit tests escritos y pasan (coverage >70%)
     ✓ Code review aprobado (≥1 revisor)
     ✓ Documentación actualizada (README, API docs)
     ✓ Testing manual completado
     ✓ Aprobación de QA (si aplica)
     ✓ Deploy a ambiente de testing exitoso
  3. Poster visible en área de trabajo
  4. Checklist en sistema de tickets

ESFUERZO: 8 horas (workshop + documentación)
IMPACTO: PPQA, inmediato

MÉTRICA DE ÉXITO:
  - 100% de tickets cumplen DoD antes de cerrar
  - Reducción de reaperturas de tickets
```

Quick Win #3: Dashboard de Métricas Básicas (Semanas 3-4)

```
PROBLEMA ACTUAL:
  Métricas dispersas, difíciles de acceder

SOLUCIÓN:
  1. Crear dashboard en Power BI o similar
  2. Métricas a incluir:
     - Tickets abiertos vs cerrados (trend)
     - Lead time promedio por tipo
     - Tickets reabiertos (%)
     - Distribución por prioridad
     - Workload por persona
     - Aging de tickets (>30 días)
  3. Actualización automática (conexión a BD)
  4. Accesible a todo el equipo

ESFUERZO: 24 horas desarrollo + integración
IMPACTO: MA, PMC, visibilidad

MÉTRICA DE ÉXITO:
  - Dashboard consultado ≥3 veces/semana por PMs
  - Decisiones basadas en datos del dashboard
```

Piloto #1: Proceso de Gestión de Requerimientos (Semanas 3-4)

```
SCOPE:
  Implementar proceso REQM completo en 1 proyecto piloto

PROYECTO PILOTO:
  Ticket tipo A (sistema nuevo) de complejidad media
  Duración estimada: 6 semanas

PROCESO A PILOTEAR:
  1. Captura de requerimientos
     - Template mejorado
     - Criterios de aceptación obligatorios
     - Priorización formal (MoSCoW)

  2. Revisión y aprobación
     - Review meeting con stakeholder
     - Sign-off documentado
     - Baseline establecida

  3. Gestión de cambios
     - Change request form
     - Análisis de impacto obligatorio
     - Aprobación formal para cambios

  4. Trazabilidad
     - Req → Design doc → User story → Code → Test case
     - Links en herramienta

  5. Validación
     - Checklist de conformidad
     - Acceptance testing
     - Cierre formal

EQUIPO PILOTO:
  - 1 PM (líder del piloto)
  - 2 desarrolladores
  - 1 QA
  - 1 analista de negocio

SOPORTE:
  - SEPG disponible para consultas
  - Retrospectiva semanal
  - Ajustes on-the-fly permitidos

ENTREGABLES:
  - Documentación del proceso refinado
  - Templates validados
  - Lecciones aprendidas
  - Métricas del piloto vs baseline
```

MES 3: ROLLOUT DE QUICK WINS Y GOVERNANCE

Semanas 1-2: Rollout de Quick Wins

```
ACTIVIDADES:
  1. Comunicar éxitos del piloto
     - Demo en all-hands
     - Newsletter interno
     - Testimoniales del equipo piloto

  2. Capacitar a equipos restantes
     - 2 sesiones × 2 hrs
     - Todos los quick wins
     - Hands-on practice

  3. Rollout progresivo
     - Week 1: Equipos A y B
     - Week 2: Equipos C y D
```

```
  4. Support desk
     - SEPG disponible para soporte
     - Slack channel #cmmi-help
     - Office hours diarias (30 min)

MÉTRICAS:
  - Adoption rate (% tickets usando nuevos procesos)
  - Time to proficiency (días hasta uso fluido)
  - Satisfaction score (encuesta post-capacitación)
```

Semanas 3-4: Establecer Governance

```
ESTRUCTURAS:
═══════════════════════════════════════════

1. SEPG (Software Engineering Process Group)
   ├─ Composición:
   │  ├─ Process Engineer (full-time)
   │  ├─ Senior Developer (20% time)
   │  └─ QA Lead (20% time)
   │
   ├─ Responsabilidades:
   │  ├─ Diseñar y actualizar procesos
   │  ├─ Capacitar al equipo
   │  ├─ Realizar auditorías
   │  ├─ Gestionar asset library
   │  └─ Reportar a dirección
   │
   └─ Reuniones:
      └─ Weekly sync (1 hr)

2. Change Control Board (para CM)
   ├─ Composición:
   │  ├─ IT Manager (chair)
   │  ├─ Tech Lead
   │  ├─ Product Owner
   │  └─ DevOps Engineer
   │
   ├─ Responsabilidades:
   │  ├─ Aprobar cambios mayores
   │  ├─ Evaluar impacto de cambios
   │  └─ Resolver conflictos
   │
   └─ Reuniones:
      └─ Bi-weekly (1 hr) o on-demand

3. Quality Assurance Team
   ├─ Composición:
   │  ├─ QA Lead
   │  └─ 2 QA Engineers
   │
   ├─ Responsabilidades:
   │  ├─ Auditorías de proceso
   │  ├─ Auditorías de producto
   │  ├─ Reportar non-conformances
   │  └─ Verificar correcciones
   │
   └─ Auditorías:
      └─ Monthly

POLÍTICAS:
  ├─ Proceso de aprobación de cambios a procesos
  ├─ Frecuencia de auditorías
  ├─ Escalación de issues críticos
  └─ Revisión de métricas con management

CALENDARIO:
  ├─ Monthly: QA audits
```

```
  ├─ Quarterly: Management review (métricas, progreso)
  ├─ Bi-annual: Full process review
  └─ Annual: Strategic planning
```

Fin de Fase 1 - Checkpoint

```
ENTREGABLES COMPLETADOS:
✓ Assessment report
✓ Implementation plan aprobado
✓ 3 Quick wins implementados
✓ 1 Piloto completado exitosamente
✓ Governance structures establecidas
✓ Equipo capacitado en fundamentos

MÉTRICAS DE ÉXITO:
✓ 80%+ adoption de quick wins
✓ Satisfaction score ≥7/10
✓ Piloto mostró ≥20% mejora en al menos 2 métricas
✓ Buy-in de management confirmado
✓ Presupuesto para Fase 2 aprobado

DECISIÓN GO/NO-GO:
  Si todas las métricas se cumplen → Proceder a Fase 2
  Si no → Retrospectiva, ajustes, repetir piloto
```

3.3 Fase 2 - CONSTRUCCIÓN (Meses 4-9)

MESES 4-5: Implementar REQM y CM (Áreas Más Fuertes)

Objetivo: Llevar REQM y CM de 60% y 55% a 85%+

REQM: Requirements Management

```
SEMANA 1-2: DISEÑO DE PROCESOS
═══════════════════════════════════════════════

Taller de Diseño de Proceso (2 días):
  Participantes: SEPG, PMs, Analistas, Stakeholders

  Output:
    1. Proceso "Gestión de Requerimientos ISSEG v1.0"
       ├─ Flowchart visual
       ├─ Descripción narrativa
       ├─ Roles y responsabilidades (RACI)
       ├─ Templates
       └─ Criterios de entrada/salida

    2. Templates:
       ├─ Requirement Specification Template
       ├─ Change Request Form
       ├─ Traceability Matrix Template
       ├─ Requirements Review Checklist
       └─ Acceptance Criteria Template

    3. Definiciones:
       ├─ Tipos de requerimientos
       ├─ Atributos obligatorios/opcionales
       ├─ Estados de requerimiento
       └─ Criterios de calidad

SEMANA 3-4: CONFIGURACIÓN DE HERRAMIENTAS
═══════════════════════════════════════════════

Jira/Azure DevOps Configuration:
```

```
  1. Custom Fields:
     ├ Requirement Type (Functional, Non-functional, Interface, Data)
     ├ Requirement Status (Proposed, Analyzed, Approved, Implemented, Verified)
     ├ Priority (Critical, High, Medium, Low)
     ├ Complexity (XS, S, M, L, XL)
     ├ Stakeholder
     ├ Related Requirements (link)
     ├ Acceptance Criteria (multi-line text)
     ├ Rationale
     └ Source (document, meeting, user feedback)

  2. Workflows:
     Proposed → In Analysis → Approved → In Dev → In Test → Verified
        ↓                        ↓          ↓
     Rejected              On Hold      Blocked

  3. Dashboards:
     ├ Requirements Coverage (% by status)
     ├ Requirements Volatility (changes/week)
     ├ Orphan Requirements (sin links)
     └ Approval Pending (aging)

  4. Automation:
     ├ Auto-assign to PM when status → "In Analysis"
     ├ Alert if no acceptance criteria after 2 days
     ├ Notify stakeholder when status → "Approved"
     └ Block transition to "Approved" if AC empty

SEMANA 5-6: CAPACITACIÓN
═══════════════════════════════════════════════

Programa de Capacitación (8 horas por persona):

  Sesión 1 (2 hrs): Fundamentos REQM
    ├ ¿Por qué es importante?
    ├ Proceso ISSEG paso a paso
    ├ Roles y responsabilidades
    └ Q&A

  Sesión 2 (2 hrs): Hands-on con Templates
    ├ Ejercicio: Capturar requerimiento
    ├ Ejercicio: Escribir acceptance criteria
    ├ Ejercicio: Crear change request
    └ Review en grupo

  Sesión 3 (2 hrs): Uso de Herramienta
    ├ Demo de Jira configurado
    ├ Práctica: Crear ticket, llenar campos
    ├ Práctica: Linking requirements
    └ Dashboard walkthrough

  Sesión 4 (2 hrs): Casos Especiales y Troubleshooting
    ├ ¿Qué hacer cuando requerimiento ambiguo?
    ├ ¿Cómo gestionar cambios urgentes?
    ├ Escalación de conflictos
    └ Final Q&A

SEMANA 7-8: PILOTO AMPLIADO
═══════════════════════════════════════════════

Scope: Todos los tickets nuevos siguen proceso REQM

Soporte:
  ├ SEPG on-call daily
  ├ Retrospectiva semanal
  └ Ajustes rápidos según feedback

Métricas a Rastrear:
  ├ % tickets con AC completos
  ├ % requirements con trazabilidad
  ├ Tiempo promedio en "Proposed" (should be <3 days)
```

```
  ├─ # change requests por requirement
  └─ Satisfaction score del equipo

Target: 80%+ compliance

SEMANA 9-10: REFINAMIENTO Y ROLLOUT COMPLETO
═══════════════════════════════════════════════

  ├─ Analizar datos del piloto
  ├─ Incorporar lecciones aprendidas
  ├─ Actualizar proceso y templates (v1.1)
  ├─ Rollout a 100% de proyectos
  └─ Primera auditoría PPQA de REQM
```

CM: Configuration Management

```
SEMANA 1-2: DISEÑO DE SISTEMA CM
═══════════════════════════════════════════════

1. Definir Configuration Items (CI):
   ├─ Código fuente (ya en Git)
   ├─ Base de datos (scripts DDL, DML, migrations)
   ├─ Documentación (requirements, design, user guides)
   ├─ Configuraciones (app configs, environment variables)
   ├─ Build scripts y pipelines
   └─ Test artifacts (test cases, test data)

2. Estrategia de Branching (Git):

   Adoptar Git Flow:

   main (production)
     ↑
   release/vX.Y
     ↑
   develop (integration)
     ↑ ↑ ↑
   feature/XXX  bugfix/YYY  hotfix/ZZZ

   Reglas:
   ├─ Feature branches de develop
   ├─ PRs obligatorios para merge a develop
   ├─ Release branches para QA
   ├─ Hotfix solo para prod crítico
   └─ Tags para cada release

3. Políticas:
   ├─ Commit messages: "TICKET-123: Description"
   ├─ PR requiere 1 aprobación (2 si crítico)
   ├─ CI debe pasar antes de merge
   ├─ No commits directos a main/develop
   └─ Code review checklist obligatorio

4. Herramientas:
   ├─ Git (ya existe)
   ├─ GitHub/GitLab/Azure Repos
   ├─ Artifact repository (Artifactory o Azure Artifacts)
   └─ CI/CD (GitHub Actions o Azure Pipelines)

SEMANA 3-4: SETUP DE INFRAESTRUCTURA CM
═══════════════════════════════════════════════

1. Configurar Repositorios:
   ├─ Migrar todo código a estructura estándar
   ├─ Agregar .gitignore apropiado
   ├─ Branch protection rules
   └─ Webhooks para integración

2. Configurar CI/CD:
```

```
   Pipeline básico:

   Commit/PR → CI Pipeline
     ├ Build
     ├ Unit Tests
     ├ Linting/Static Analysis (SonarQube)
     ├ Security Scan
     └ Artifact Generation

   Release → CD Pipeline
     ├ Deploy to Dev (automático)
     ├ Deploy to QA (automático)
     ├ Deploy to Staging (click to deploy)
     └ Deploy to Prod (approval required)

3. Artifact Repository:
   ├ Setup Artifactory o similar
   ├ Versionamiento semántico (vX.Y.Z)
   ├ Retention policy (guardar últimas 10 versiones)
   └ Traceability (artifact → commit → ticket)

4. Baseline Definitions:
   ├ Development baseline: Tag en develop cada sprint
   ├ Release baseline: Release branch + tag
   ├ Production baseline: Tag en main al deployar
   └ Documentar en release notes

SEMANA 5-6: CAPACITACIÓN CM
═══════════════════════════════════════════════

Programa de Capacitación (6 horas por persona):

  Sesión 1 (2 hrs): Git Workflow
    ├ Git Flow explicado
    ├ Branching strategy
    ├ Commit best practices
    └ Hands-on: Crear feature branch, commit, PR

  Sesión 2 (2 hrs): CI/CD Pipeline
    ├ ¿Qué es CI/CD?
    ├ Pipeline ISSEG walkthrough
    ├ Cómo interpretar resultados
    └ Troubleshooting común

  Sesión 3 (2 hrs): Change Control
    ├ Cuándo crear Change Request
    ├ Change Control Board explicado
    ├ Proceso de aprobación
    └ CM audits

SEMANA 7-8: IMPLEMENTACIÓN
═══════════════════════════════════════════════

  ├ Migrar todos los proyectos a nuevo workflow
  ├ Habilitar branch protection
  ├ Activar CI/CD pipelines
  └ Primer release usando proceso formal

SEMANA 9-10: CONSOLIDACIÓN
═══════════════════════════════════════════════

  ├ Primera reunión Change Control Board
  ├ Auditoría CM por PPQA
  ├ Refinamientos basados en experiencia
  └ Documentar lecciones aprendidas

MÉTRICAS DE ÉXITO:
  ✓ 100% código en Git con workflow estándar
  ✓ 100% PRs tienen review antes de merge
  ✓ CI pipeline pasa en ≥95% de builds
```

```
  ✓ 0 commits directos a main/develop
  ✓ Release process documentado y seguido
  ✓ CM audits con ≥85% compliance
```

Fin Mes 5 - Milestone Check

```
REQM Target: 85% → Alcanzado: ____%
CM Target: 85% → Alcanzado: ____%

Si ambos ≥80%: Proceder a siguiente conjunto
Si alguno <80%: Extender 2-4 semanas adicionales
```

MESES 6-7: Implementar PP y PMC

PP: Project Planning

```
SEMANA 1-2: DISEÑO DE PROCESO DE PLANIFICACIÓN
═══════════════════════════════════════════════

Workshop de Planificación (1.5 días):

  Output:
    1. Proceso "Planificación de Proyectos ISSEG v1.0"
    2. Templates:
       ├─ Project Charter Template
       ├─ Work Breakdown Structure (WBS) Template
       ├─ Estimation Template (story points + hours)
       ├─ Resource Allocation Plan
       ├─ Risk Register Template
       └─ Project Schedule Template

    3. Modelo de Estimación:
       Definir método de estimation para ISSEG:

       Opción A: Story Points (Fibonacci):
       1, 2, 3, 5, 8, 13, 21

       Calibración:
       - 1 point = ~2 hours (muy simple, sin incertidumbre)
       - 3 points = ~1 day (simple, bien entendido)
       - 5 points = ~2-3 days (complejidad moderada)
       - 8 points = ~1 semana (complejo)
       - 13+ points = Debe descomponerse

       Factores de ajuste:
       - Complejidad técnica: ×1.0 a ×1.5
       - Incertidumbre de req: ×1.0 a ×1.3
       - Dependencies: ×1.0 a ×1.2
       - Team experience: ×0.8 a ×1.2

    4. Criterios de Proyectos:
       ├─ Pequeño: <40 story points, 1-2 personas, <1 mes
       ├─ Mediano: 40-100 points, 2-4 personas, 1-3 meses
       ├─ Grande: >100 points, >4 personas, >3 meses

       Nivel de planificación por tamaño:
       - Pequeño: Plan ligero (2 páginas)
       - Mediano: Plan estándar (5-10 páginas)
       - Grande: Plan completo (15+ páginas)

SEMANA 3: CONFIGURACIÓN DE HERRAMIENTAS
═══════════════════════════════════════

1. Configurar Project Management Tool:
```

```
   Opción: Jira + Advanced Roadmaps /o Azure Boards

   Setup:
   ├─ Jerarquía:
   │  Epic (proyecto grande)
   │    ↓
   │  Story/Ticket (feature)
   │    ↓
   │  Sub-task (tareas técnicas)
   │
   ├─ Campos adicionales:
   │  ├─ Story Points
   │  ├─ Original Estimate (hours)
   │  ├─ Time Spent
   │  ├─ Remaining Estimate
   │  ├─ Risk Level
   │  └─ Dependencies (links)
   │
   └─ Views:
      ├─ Backlog
      ├─ Sprint Board (Kanban/Scrum)
      ├─ Roadmap (Gantt-style)
      └─ Portfolio view

2. Dashboard de Proyecto:
   ├─ Burndown chart (si Scrum)
   ├─ Cumulative flow diagram
   ├─ Velocity trend
   ├─ EVM metrics (si proyecto grande):
   │  ├─ PV (Planned Value)
   │  ├─ EV (Earned Value)
   │  ├─ AC (Actual Cost)
   │  ├─ SPI (Schedule Performance Index)
   │  └─ CPI (Cost Performance Index)
   ├─ Risk heatmap
   └─ Resource allocation

SEMANA 4: CAPACITACIÓN PP
═══════════════════════════════════════════════

Programa (6 horas):

  Sesión 1 (2 hrs): Fundamentos de Planificación
    ├─ Importancia de planificar
    ├─ Proceso ISSEG step-by-step
    ├─ Roles (quién planifica qué)
    └─ Templates walkthrough

  Sesión 2 (2 hrs): Estimation Techniques
    ├─ Story points explicados
    ├─ Planning poker (ejercicio)
    ├─ Factores de ajuste
    ├─ Usar datos históricos
    └─ Práctica con tickets reales

  Sesión 3 (2 hrs): Risk Management Basics
    ├─ Identificar riesgos
    ├─ Probabilidad × Impacto
    ├─ Estrategias de mitigación
    └─ Crear risk register

SEMANAS 5-8: IMPLEMENTACIÓN GRADUAL
═══════════════════════════════════════════════

Week 5-6: Pilotos con 2 proyectos medianos
  ├─ Crear Project Charter
  ├─ Estimation session (todo el equipo)
  ├─ WBS y cronograma
  ├─ Risk register
  └─ Revisión con stakeholders
```

```
Week 7-8: Rollout a todos los proyectos nuevos
  ├─ Todo proyecto nuevo requiere plan
  ├─ SEPG revisa planes
  ├─ Template refinements según feedback
  └─ Métricas: % proyectos con plan, calidad de estimaciones
```

PMC: Project Monitoring & Control

```
SEMANA 1-2: DISEÑO DE PROCESO DE MONITOREO
═══════════════════════════════════════════

Workshop (1 día):

  Output:
    1. Proceso "Monitoreo y Control ISSEG v1.0"
    2. Métricas de Proyecto Estándar:
       ├─ Schedule metrics:
       │  ├─ Actual vs Planned progress
       │  ├─ SPI (Schedule Performance Index)
       │  └─ Variance análisis
       ├─ Quality metrics:
       │  ├─ Defect density
       │  ├─ Defect leakage rate
       │  └─ Rework %
       ├─ Productivity:
       │  ├─ Velocity (points/sprint)
       │  ├─ Throughput (tickets/week)
       │  └─ Cycle time
       └─ Resource utilization:
          ├─ Allocation %
          ├─ Availability
          └─ Overtime hours

    3. Frecuencia de Revisiones:
       ├─ Daily: Standup (15 min) - equipo
       ├─ Weekly: Status review (30 min) - PM + Tech Lead
       ├─ Bi-weekly: Sprint review (1 hr) - equipo + stakeholders
       ├─ Monthly: Project review (1 hr) - PM + Management
       └─ Quarterly: Portfolio review (2 hrs) - C-level

    4. Umbrales de Alerta (Thresholds):
       🟢 Green: SPI/CPI > 0.95, on track
       🟡 Yellow: SPI/CPI 0.85-0.95, caution
       🔴 Red: SPI/CPI < 0.85, action required

    5. Escalation Process:
       Issue identified
         ↓
       PM attempts resolution (2 days)
         ↓ (si no resuelto)
       Escalate to Manager (3 days)
         ↓ (si no resuelto)
       Escalate to Steering Committee
         ↓
       Decision & action plan

SEMANA 3: CONFIGURACIÓN DE DASHBOARDS Y ALERTAS
═══════════════════════════════════════════

1. Dashboards Automáticos:

   Dashboard de Proyecto Individual:
   ├─ RAG Status (Red/Amber/Green)
   ├─ Progress (%complete vs target)
   ├─ Metrics (SPI, CPI, velocity, defects)
   ├─ Risk status (# high risks)
   ├─ Top issues (bloqueadores)
   └─ Upcoming milestones
```

```
   Dashboard de Portfolio:
   ├─ Todos los proyectos (1 línea c/u)
   ├─ Filtros: Por estado, por PM, por área
   ├─ Capacity view (recursos vs demanda)
   └─ Health score agregado

2. Alertas Automáticas:
   ├─ Ticket aging >21 días sin movimiento
   ├─ Sprint burndown no healthy
   ├─ Velocity drop >20% vs promedio
   ├─ High/critical defects sin asignar >24 hrs
   └─ Budget overrun >10%

SEMANA 4: CAPACITACIÓN PMC
═══════════════════════════════════════════════

Programa (4 horas):

  Sesión 1 (2 hrs): Monitoring Essentials
    ├─ Qué monitorear y por qué
    ├─ Interpretar dashboards
    ├─ RAG status criteria
    └─ Hands-on: Analizar dashboard de ejemplo

  Sesión 2 (2 hrs): Control & Corrective Actions
    ├─ Identificar varianzas
    ├─ Root cause analysis ligero
    ├─ Corrective vs preventive actions
    ├─ Escalation process
    └─ Case studies

SEMANAS 5-8: IMPLEMENTACIÓN
═══════════════════════════════════════════════

Week 5-6: Setup de revisiones
  ├─ Calendario de revisiones definido
  ├─ Templates de status report
  ├─ Primeras revisiones semanales
  └─ Ajustes según feedback

Week 7-8: Full adoption
  ├─ Dashboards actualizados automáticamente
  ├─ Métricas revisadas en reuniones
  ├─ Acciones correctivas rastreadas
  └─ Primera auditoría PMC

MÉTRICAS DE ÉXITO:
  ✓ 100% proyectos con plan
  ✓ 100% proyectos con dashboard actualizado
  ✓ Status reviews realizadas ≥95% del tiempo
  ✓ Issues resueltos en <5 días promedio
  ✓ Varianzas identificadas proactivamente (no reactivamente)
```

MESES 8-9: Implementar MA y PPQA

MA: Measurement & Analysis

```
SEMANA 1-2: DISEÑO DE PROGRAMA DE MEDICIÓN
═══════════════════════════════════════════════

Workshop de Métricas (2 días):

  Output:
    1. "Programa de Medición ISSEG v1.0"

       Objetivos de Medición:
       ├─ Objetivo 1: Mejorar predictibilidad de entregas
       │  ├─ Pregunta: ¿Cumplimos cronograma?
```

```
        │  └─ Métricas: SPI, Lead Time, On-time delivery %
        │
        ├─ Objetivo 2: Mejorar calidad del producto
        │  ├─ Pregunta: ¿Entregamos con pocos defectos?
        │  └─ Métricas: Defect density, Leakage rate, MTTR
        │
        ├─ Objetivo 3: Optimizar productividad
        │  ├─ Pregunta: ¿Somos eficientes?
        │  └─ Métricas: Velocity, Throughput, Effortvariance
        │
        └─ Objetivo 4: Aumentar satisfacción
           ├─ Pregunta: ¿Clientes y equipo felices?
           └─ Métricas: CSAT, eNPS, Team morale

    2. Measurement Specifications (por cada métrica):

       Ejemplo: "Lead Time"
       ├─ Definición operacional:
       │  Tiempo desde "Ticket creado" hasta "Deployed to Production"
       ├─ Unidad: Días calendario
       ├─ Frecuencia de recolección: Continua (al cerrar ticket)
       ├─ Fuente de datos: Sistema de tickets (automated)
       ├─ Responsable: Automated script
       ├─ Visualización: Histogram + median trend
       ├─ Target: Mediana ≤21 días (tipo C), ≤45 días (tipo B)
       └─ Uso: Dashboard ejecutivo, retrospectivas

    3. Especificar para 15 métricas core

SEMANA 3: SETUP DE DATA WAREHOUSE
═══════════════════════════════════════════════════

1. Data Warehours architecture:

   Operational DBs → ETL → Data Warehouse → BI Tool
   (Jira, Git, etc)        (SQL Server)     (Power BI)

2. ETL Process:
   ├─ Extract nightly from sources
   ├─ Transform to standardized schema
   ├─ Load to DW
   └─ Data quality checks

3. Star Schema Design:

   Fact Tables:
   ├─ Fact_Tickets (tickets completados)
   ├─ Fact_Defects (bugs reportados)
   └─ Fact_Efforts (time tracking)

   Dimension Tables:
   ├─ Dim_Date
   ├─ Dim_Team_Member
   ├─ Dim_Project
   ├─ Dim_Ticket_Type
   └─ Dim_Priority

SEMANA 4: DASHBOARDS ANALÍTICOS
═══════════════════════════════════════════════════

Crear 3 niveles de dashboards:

1. Operational Dashboard (Para equipo):
   ├─ Real-time metrics
   ├─ Today's focus
   ├─ Blockers y issues
   └─ Refresh: Cada hora

2. Tactical Dashboard (Para PMs):
   ├─ Project health
   ├─ Trends últimas 4 semanas
```

```
   ├─ Comparativa entre proyectos
   └─ Refresh: Diario

3. Strategic Dashboard (Para Ejecutivos):
   ├─ KPIs agregados
   ├─ Trends trimestrales
   ├─ Benchmarks (internos y externos)
   └─ Refresh: Semanal

SEMANAS 5-6: CAPACITACIÓN Y ROLLOUT
═══════════════════════════════════════════════

Capacitación (4 horas):
  ├─ Importancia de métricas
  ├─ Interpretación de dashboards
  ├─ Uso de datos para decisiones
  └─ Evitar gaming de métricas

Rollout:
  ├─ Dashboards disponibles a todos
  ├─ Primeras decisiones data-driven
  └─ Retrospectiva sobre métricas

SEMANAS 7-8: ANÁLISIS Y OPTIMIZACIÓN
═══════════════════════════════════════════════

  ├─ Análisis de tendencias (primeros 2 meses de datos)
  ├─ Identificar áreas de mejora con datos
  ├─ Ajustar targets si necesario
  └─ Plan de acción basado en insights
```

PPQA: Process & Product Quality Assurance

```
SEMANA 1: DISEÑO DE PROGRAMA QA
═══════════════════════════════════════════════

Workshop QA (1 día):

  Output:
    1. "Plan de Aseguramiento de Calidad ISSEG v1.0"

    2. Auditorías de Proceso:
       Frecuencia: Mensual
       Sample: 20% de proyectos activos (mínimo 2)

       Checklist de Auditoría (por área):

       REQM:
       ├─ ¿Requerimientos capturados en sistema? ✓/✗
       ├─ ¿Criterios de aceptación presentes? ✓/✗
       ├─ ¿Aprobaciones documentadas? ✓/✗
       ├─ ¿Trazabilidad a diseño/código? ✓/✗
       └─ ¿Cambios gestionados formalmente? ✓/✗

       (Similar para PP, PMC, CM, MA)

       Scoring: (# ✓ / total) × 100 = % compliance

       Target: ≥85% compliance

    3. Revisiones de Producto:

       Code Reviews (ya en CM):
       ├─ Obligatorio para todo merge
       ├─ Checklist de código (estándares)
       └─ Security checklist

       Design Reviews:
       ├─ Para proyectos medianos/grandes
```

```
        ├─ Antes de inicio de desarrollo
        ├─ Participantes: Architect, Tech Lead, PM
        └─ Criterios: Arquitectura, escalabilidad, performance

        Testing:
        ├─ Unit test coverage ≥70%
        ├─ Integration testing obligatorio
        ├─ UAT con checklist
        └─ Regression testing en releases

    4. Issue Management:

        Non-Conformances (NCs):
        ├─ Detectado en auditoría → NC logged
        ├─ Severidad asignada (Major, Minor)
        ├─ Owner asignado
        ├─ Corrective action definida
        ├─ Timeline para resolución (7-14 días)
        └─ Verificación de cierre

        Trending:
        ├─ # NCs por mes (debería bajar)
        ├─ Areas con más NCs (focus de mejora)
        └─ Repeat NCs (señal de problema sistémico)

SEMANA 2: CAPACITACIÓN AUDITORES
═══════════════════════════════════════════════

Capacitar a 3 personas como auditores internos:

  Contenido (8 horas):
  ├─ Fundamentos de auditorías
  ├─ Uso de checklists
  ├─ Entrevistar sin ser confrontacional
  ├─ Evaluar evidencia objetivamente
  ├─ Escribir findings claros
  └─ Follow-up de correcciones

SEMANAS 3-4: PRIMERAS AUDITORÍAS
═══════════════════════════════════════════════

  ├─ Auditar 2 proyectos piloto
  ├─ Generar reporte de hallazgos
  ├─ Presentar a equipo (sin culpar)
  ├─ Plan de corrección
  └─ Re-auditar después de correcciones

SEMANAS 5-6: ROLLOUT COMPLETO
═══════════════════════════════════════════════

  ├─ Calendario de auditorías todo el año
  ├─ Proceso documentado y comunicado
  ├─ Dashboard de PPQA (compliance trends)
  └─ Integración con meetings de management

SEMANAS 7-8: CONSOLIDACIÓN
═══════════════════════════════════════════════

  ├─ Análisis de tendencias (primeras auditorías)
  ├─ Identificar pain points sistémicos
  ├─ Mejoras de proceso basadas en NCs
  └─ First "clean audit" (meta: 90%+ compliance)

MÉTRICAS DE ÉXITO:
  ✓ Auditorías realizadas según calendario (100%)
  ✓ Compliance promedio ≥80% al final de Mes 9
  ✓ NCs resueltos en <14 días (95%)
  ✓ Tendencia de compliance: Creciente
  ✓ Acceptance del equipo (no visto como policía)
```

Fin de Fase 2 - Checkpoint

```
TODAS LAS ÁREAS NIVEL 2 IMPLEMENTADAS:

REQM  ████████████▒▒  85% ✓
PP    ████████████▒▒  82% ✓
PMC   ████████████▒▒  81% ✓
MA    ████████████▒▒  80% ✓
PPQA  ██████████▒▒▒▒  78% ~ (aceptable)
CM    ████████████▒▒  87% ✓

PROMEDIO: 82% → On track para Nivel 2

Decisión: Proceder a Fase 3 (Consolidación)
```

3.4 Fase 3 - CONSOLIDACIÓN (Meses 10-15)

MESES 10-12: Integración y Optimización

Objetivo: Hacer que procesos funcionen como sistema cohesivo

```
MES 10: INTEGRACIÓN DE PROCESOS
═══════════════════════════════════════════════

Actividades:

  1. Mapear Flujo End-to-End:

     Nuevo Sistema (Ticket Tipo A):

     Requirements → Planning → Monitoring → Development → QA → Release
         (REQM)       (PP)        (PMC)                  (PPQA)  (CM)
                                   ↓ [continuous]
                                Measurement (MA)

     Documentar cada handoff:
     ├─ Qué se entrega (artifacts)
     ├─ Criterios de aceptación
     ├─ Roles responsables
     └─ Herramientas usadas

  2. Eliminar Redundancias:
     - ¿Estamos pidiendo misma info múltiples veces?
     - ¿Hay pasos innecesarios?
     - ¿Dónde hay cuellos de botella?

  3. Optimizar Herramientas:
     - Integrar Jira con Git (automatic linking)
     - Dashboard consolidado (una fuente de verdad)
     - Automatizar todo lo posible

  4. Refinar Templates:
     - Versión 2.0 de cada template
     - Basado en 6 meses de uso
     - Más ligeros, más útiles

  5. Actualizar Capacitación:
     - Onboarding package para nuevos miembros
     - Refresher para equipo actual (medio día)
     - Advanced topics (opcional)

MES 11: OPTIMIZACIÓN BASADA EN DATOS
═══════════════════════════════════════════════

Análisis Profundo:
```

```
  1. Revisar 6 meses de metrics:
     ├─ ¿Qué ha mejorado? (celebrar)
     ├─ ¿Qué no ha mejorado? (analizar por qué)
     ├─ ¿Dónde estamos estancados?
     └─ ¿Hay efectos inesperados (buenos o malos)?

  2. Benchmarking:
     ├─ Interno: Equipos entre sí
     ├─ Externo: Industria (si disponible)
     └─ Identificar "best performers" internos

  3. Experimentos de Mejora:
     - Identificar 3 hipótesis de mejora
     - Ejemplo: "Si reducimos Work-in-Progress a 3 tickets/persona,
                 el cycle time bajará 20%"
     - Pilotear 4 semanas
     - Medir resultados
     - Adoptar o descartar

  4. Kaizen Events:
     - 2 días de "improvement blitz"
     - Todo el equipo participa
     - Identificar y resolver pain points pequeños
     - Target: 10-15 mejoras implementadas

MES 12: PREPARACIÓN PARA EVALUACIÓN
═══════════════════════════════════════════════

Si se busca certificación formal (o solo validación):

  1. Mini-Assessment Interno:
     ├─ SEPG realiza assessment completo
     ├─ Identificar gaps finales
     ├─ Plan de cierre de gaps (2-3 meses)
     └─ Estimar readiness (% por área)

  2. Auditoría Exhaustiva:
     ├─ Revisar 100% de proyectos activos
     ├─ Verificar evidencia para todas las prácticas
     ├─ Documentar en "Evidence Binder"
     └─ Correcciones si necesario

  3. Dry-Run Evaluation:
     ├─ Contratar consultor externo (opcional)
     ├─ Simular evaluación SCAMPI
     ├─ Obtener feedback objetivo
     └─ Últimos ajustes

  4. Decisión Go/No-Go para evaluación formal:
     ├─ Todas áreas ≥85%: GO
     ├─ Alguna <80%: Posponer, trabajar 2-3 meses más
     └─ Si no se busca certificación: Validación interna

Entregables Mes 12:
  ✓ Procesos refinados v2.0
  ✓ Métricas mostrando mejora sostenida
  ✓ Evidence binder completo
  ✓ Equipo confiado y competente
  ✓ Decisión sobre evaluación formal
```

MESES 13-15: Sostenibilidad y Mejora Continua

```
MES 13-14: INSTITUCIONALIZAR CMMI
═══════════════════════════════════════════════

Hacer que CMMI sea "business as usual":

  1. Actualizar Job Descriptions:
```

```
      - Incluir responsabilidades de proceso
      - Parte de performance reviews
      - Incentivos alineados con compliance

  2. Integrar en Onboarding:
      - Nuevos miembros capacitados en Semana 1
      - Buddy system (assigned mentor)
      - Checklist de competencias

  3. Comunidad de Práctica:
      - Monthly "Lunch & Learn"
      - Show & tell de proyectos exitosos
      - Compartir lecciones aprendidas

  4. Biblioteca de Activos:
      - Process Asset Library (PAL)
      - Todos los templates centralizados
      - Lecciones aprendidas catalogadas
      - Examples y case studies internos
      - Fácilmente searchable (wiki o Confluence)

  5. Mantener Momentum:
      - Quarterly all-hands: Mostrar métricas y éxitos
      - Reconocimiento: "Process Champion of the Quarter"
      - Newsletter mensual de mejora de procesos

MES 15: EVALUACIÓN FORMAL (Si se decidió hacerla)
═══════════════════════════════════════════════

SCAMPI Class A (Benchmark Quality):

  Preparación (2-3 semanas antes):
  ├─ Briefing del equipo
  ├─ Evidence organizada y accesible
  ├─ Cronograma de entrevistas
  └─ Facilities preparadas

  Evaluación (1 semana):
  ├─ Día 1: Introduction, document review
  ├─ Día 2-3: Entrevistas y observación
  ├─ Día 4: Consolidación del equipo evaluador
  ├─ Día 5: Presentación de hallazgos
  └─ Resultado: Nivel alcanzado (1, 2, 3, 4, 5)

  Post-Evaluación:
  ├─ Celebración (si exitoso)
  ├─ Plan de cierre de gaps (si hubo)
  ├─ Comunicación de logro (marketing interno/externo)
  └─ Appraisal record preservation

O, si No Evaluación Formal:

  Validación Interna Rigurosa:
  ├─ Assessment por SEPG + Consultor externo
  ├─ Reporte de nivel alcanzado (estimado)
  ├─ Roadmap hacia Nivel 3 (si aplicable)
  └─ Continuar mejora continua

Fin de Fase 3 - LOGRO CELEBRADO
═══════════════════════════════

🎉 NIVEL 2 ALCANZADO (85%+ en todas las áreas)

Próximos pasos:
  ├─ Mantener compliance (no regresar)
  ├─ Co ntinuar mejora continua
  ├─ Considerar Nivel 3 (Año 2-3)
  └─ Compartir historia de éxito
```

4. Desarrollo Paso a Paso por Área de Proceso

(Sección completa con instrucciones detalladas para implementar cada área)

4.1 REQM - Requirements Management (Detalle Extremo)

Step 1: Captura de Requerimientos

Template: Specification de Requerimiento

```
## REQUERIMIENTO: [ID] - [Nombre Corto]

### 1. INFORMACIÓN GENERAL
- **ID**: REQ-ISSEG-[YYYY]-[###]
- **Proyecto**: [Nombre del proyecto/sistema]
- **Tipo de Ticket**: [A/B/C/Problema]
- **Solicitante**: [Nombre, área]
- **Fecha de Solicitud**: [YYYY-MM-DD]
- **Prioridad**: [Crítica/Alta/Media/Baja]
- **Estado**: [Propuesto/En Análisis/Aprobado/En Desarrollo/Verificado]

### 2. DESCRIPCIÓN DEL REQUERIMIENTO

#### 2.1 Descripción General
[Descripción en lenguaje natural, comprensible para stakeholders]

**Ejemplo**:
"El sistema ISSEG debe permitir a los usuarios del área de Recursos
Humanos ver los tickets asignados a su área en una vista de calendario,
donde cada ticket aparece en la fecha de su vencimiento. El calendario
debe permitir cambiar entre vista mensual y semanal."

#### 2.2 Contexto y Justificación
**¿Por qué es necesario este requerimiento?**

[Explicar el problema que resuelve o la oportunidad que aprovecha]

**Ejemplo**:
"Actualmente, RH debe revisar manualmente la lista de tickets y sus
fechas de vencimiento una por una. Esto consume ~30 minutos diarios
y ocasionalmente resulta en tickets olvidados. Una vista de calendario
reduciría este tiempo a <5 minutos y eliminaría olvidos."

#### 2.3 Tipo de Requerimiento
☐ Funcional (capacidad o comportamiento del sistema)
☐ No Funcional (calidad, restricción, performance)
☐ Interfaz (interacción con otro sistema o usuario)
☐ Dato (estructura de información)

#### 2.4 Clasificación
- **Categoría**: [UI, Backend, Integración, Infraestructura, Seguridad, etc.]
- **Módulo Afectado**: [Gestión de Tickets, Dashboard, Reportes, etc.]
- **Complejidad Inicial**: [XS/S/M/L/XL] _(refinado en planning)_

### 3. CRITERIOS DE ACEPTACIÓN

_(Usar formato Given-When-Then o checklist, lo que sea más claro)_

**AC1: Vista de Calendario Mensual**
- **Given**: Usuario de RH ha iniciado sesión
- **When**: Navega a "Mis Tickets" y selecciona vista "Calendario Mes"
- **Then**:
  ✓ Se muestra calendario del mes actual
  ✓ Cada ticket del área aparece en fecha de vencimiento
  ✓ Ticket muestra: ID, título (truncado a 40 chars), prioridad (color)
  ✓ Click en ticket abre detalle

**AC2: Vista de Calendario Semanal**
```

- **Given**: Usuario en vista de calendario
- **When**: Selecciona "Vista Semana"
- **Then**:
  ✓ Muestra 7 días completos (lun-dom)
  ✓ Misma información que vista mensual
  ✓ Más espacio vertical (muestra más info del ticket)

AC3: Navegación entre Meses/Semanas
- **Given**: Usuario en cualquier vista de calendario
- **When**: Click en botones "Anterior" o "Siguiente"
- **Then**:
  ✓ Calendario actualiza a periodo correspondiente
  ✓ Tickets de ese periodo se cargan (max 2 segundos)

AC4: Performance
- **Given**: Área con hasta 200 tickets activos
- **When**: Carga vista de calendario
- **Then**:
  ✓ Página carga en ≤3 segundos (WiFi estándar)
  ✓ No hay flickering o lag al cambiar de vista

AC5: Responsive Design
- **Given**: Usuario accede desde dispositivo móvil
- **When**: Navega a calendario
- **Then**:
  ✓ Vista se adapta a pantalla pequeña
  ✓ Usable sin scroll horizontal

4. RESTRICCIONES Y DEPENDENCIAS

4.1 Restricciones Técnicas
- [Librería de calendario: FullCalendar.js o similar]
- [Backend debe exponer API GET /api/tickets/by-area/{area_id}?start_date=...&end_date=...]
- *[Compatible con navegadores: Chrome, Firefox, Edge (últimas 2 versiones)]*

4.2 Dependencias de Otros Requerimientos
- ***Depende de**: REQ-ISSEG-2026-012 (API de filtrado de tickets por área)*
- ***Prerequisito de**: REQ-ISSEG-2026-045 (Notificaciones de tickets próximos a vencer)*

4.3 Restricciones Normativas o de Negocio
- *[Usuarios solo ven tickets de su propia área (seguridad)]*
- *[Datos sensibles no deben aparecer en vista de calendario]*

5. ESTIMACIÓN Y PLANIFICACIÓN

5.1 Estimación Inicial
- ***Story Points**: [5]* _(a refinar en planning poker)_
- **Esfuerzo Estimado**: [~1 semana, 1 desarrollador]
- **Riesgo**: [Bajo: Librería conocida, API ya existe]

5.2 Tareas Técnicas Anticipadas
*_(No WBS completo, solo primeras ideas)_*
1. Investigar y seleccionar librería de calendario (2 hrs)
2. Diseño de UI/UX (mockup) (4 hrs)
3. Implementar integración con API (8 hrs)
4. Diseño responsive (4 hrs)
5. Testing manual (4 hrs)
6. Unit tests (4 hrs)

5.3 Sprint Tentativo
- **Target Sprint**: Sprint 12 (Jun 15-28, 2026)
- **Razón**: Depende de API que se completa en Sprint 11

6. TRAZABILIDAD

6.1 Origen
- **Fuente**: Solicitud directa de Jefa de RH en reunión del 2026-02-10
- **Documentos Relacionados**:
  - [Meeting Minutes - Reunión RH 2026-02-10]
  - [Email de follow-up con requerimientos de RH]

```
#### 6.2 Relacionado Con (Links)
- **Similar a**: REQ-ISSEG-2025-089 (Vista de tabla de tickets, mismo módulo)
- **Bloqueado por**: REQ-ISSEG-2026-012 (API de filtrado)
- **Es prerequisito de**: REQ-ISSEG-2026-045 (Notificaciones)

#### 6.3 Trazabilidad a Implementación (se completa después)
- **Diseño**: [Link a diseño en Figma/Sketch]
- **User Stories**: [JIRA-1234, JIRA-1235]
- **Commits**: [Link a PR en GitHub]
- **Test Cases**: [TC-Cal-001, TC-Cal-002, TC-Cal-003]

### 7. APROBACIÓN

#### 7.1 Revisiones
| Fecha | Revisor | Rol | Comentarios | Aprobado |
|-------|---------|-----|-------------|----------|
| 2026-02-12 | Juan Pérez | Analista | Aclaró ACs, agregó AC5 | ✓ |
| 2026-02-13 | María López | Jefa de RH | Confirma que cumple necesidad | ✓ |
| 2026-02-14 | Carlos Tech Lead | Arq. Software | Validó viabilidad técnica | ✓ |

#### 7.2 Aprobación Final
- **Aprobado Por**: [María López, Product Owner]
- **Fecha de Aprobación**: [2026-02-14]
- **Firma/Email**: [Adjunto: email de aprobación]

#### 7.3 Baseline
Este requerimiento es parte de:
- **Baseline**: Baseline-v1.5-Mejoras-Dashboard
- **Fecha de Baseline**: 2026-02-15

### 8. GESTIÓN DE CAMBIOS

#### 8.1 Historial de Cambios
| Versión | Fecha | Autor | Cambios Realizados |
|---------|-------|-------|--------------------|
| 1.0 | 2026-02-11 | Ana Dev | Creación inicial |
| 1.1 | 2026-02-12 | Ana Dev | Agregado AC5 (responsive) por feedback de Juan |
| 1.2 | 2026-02-13 | Ana Dev | Aclarado constraint de seguridad |

#### 8.2 Cambios Posteriores a Aprobación
_(Requieren Change Request formal)_

[Ninguno aún]

### 9. VALIDACIÓN Y CIERRE

#### 9.1 Criterios de Validación
_(Cómo se probará que está completo)_
- ☐ Todos los ACs pasan acceptance testing
- ☐ Code review aprobado
- ☐ Unit test coverage ≥70%
- ☐ UAT con usuario de RH exitoso
- ☐ Performance validado (load testing)
- ☐ Documentación actualizada (user guide)

#### 9.2 Resultado de Validación
- **Validado Por**: [Nombre, QA Lead]
- **Fecha**: [YYYY-MM-DD]
- **Test Report**: [Link]
- **Estado Final**: [Verificado/Aprobado para Deploy]

---

**Notas**:
- Template largo intencionalmente (no todos los campos obligatorios siempre)
- Para requerimientos pequeños (XS, S), simplificar
- Para requerimientos grandes (XL), este nivel de detalle es crítico
```

Step 2: Proceso de Revisión y Aprobación

Workflow de Aprobación:

```
1. REQ capturado por Analista/PM
   ├─ Estado: "Propuesto"
   └─ Asignado a: PM

2. PM revisa completitud (24-48 hrs)
   ├─ ¿Todos campos llenos?
   ├─ ¿ACs claros y testables?
   ├─ ¿Estimación razonable?
   │
   Si NO completo:
   │   └─ Regresa a solicitante para aclaración
   │
   Si completo:
       └─ Estado → "En Análisis"
           └─ Asignado a: Tech Lead

3. Tech Lead evalúa viabilidad técnica (2-3 días)
   ├─ ¿Es factible con arquitectura actual?
   ├─ ¿Identifica dependencies?
   ├─ ¿Estimación correcta?
   │
   Si tiene issues técnicos:
   │   └─ Feedback a PM, ajusta REQ
   │
   Si viable:
       └─ Estado → "Listo para Aprobación"
           └─ Asignado a: Product Owner

4. Product Owner toma decisión de negocio (1-2 días)
   ├─ ¿Vale la pena el esfuerzo?
   ├─ ¿Prioridad alta vs otros REQs?
   ├─ ¿Budget disponible?
   │
   Si NO aprobado:
   │   └─ Estado → "Rechazado"
   │       └─ Razón documentada
   │
   Si aprobado:
       └─ Estado → "Aprobado"
           ├─ Agregado a Product Backlog
           └─ Priorizado vs otros REQs

5. Incluido en Sprint Planning
   └─ Estado → "En Desarrollo" (cuando Sprint inicia)

6. Implementado y Testeado
   └─ Estado → "En Pruebas"

7. UAT Exitoso
   └─ Estado → "Verificado"

8. Deployed a Producción
   └─ Estado → "Completado/Cerrado"
```

SLAs de Revisión:

- PM review: ⩽ 48 horas
- Tech Lead review: ⩽3 días laborales
- PO approval: ⩽ 2 días laborales
- **Total: Máximo 1 semana desde captura hasta decisión**

Step 3: Gestión de Cambios en Requerimientos

Cuando un REQ ya aprobado necesita cambio:

```
Change Request Process:
═══════════════════════════════════════════════

1. IDENTIFICACIÓN:
   ├─ Alguien identifica necesidad de cambio
   └─ Crear "Change Request" (CR) en sistema

2. DOCUMENTAR CR:

   Template de Change Request:

   ## CHANGE REQUEST: CR-[ID]

   ### Requerimiento Afectado:
   - ID: REQ-ISSEG-2026-034
   - Nombre: [Nombre del REQ]
   - Estado Actual: [En Desarrollo / Aprobado / etc.]

   ### Cambio Solicitado:
   [Descripción del cambio deseado]

   **Razón del Cambio**:
   [¿Por qué es necesario? ¿Qué descubrimos?]

   ### Análisis de Impacto:

   **Alcance**:
   ☐ Cambio menor (ajuste cosmético, no afecta funcionalidad)
   ☐ Cambio moderado (modifica ACs, +/- 20% esfuerzo)
   ☐ Cambio mayor (altera sustancialmente el REQ, >40% esfuerzo)

   **Impacto en Cronograma**:
   - Esfuerzo adicional estimado: [X días/horas]
   - Retraso en entrega: [Y días]

   **Impacto en Costo**:
   - Costo adicional: [$Z] o [X horas × tasa]

   **Impacto en Otros REQs**:
   - REQs afectados: [Lista de IDs]
   - Impacto: [Descripción]

   **Impacto en Riesgo**:
   - Nuevos riesgos introducidos: [Descripción]

   ### Alternativas Consideradas:
   1. [Opción 1: Descripción, pros/cons]
   2. [Opción 2: Descripción, pros/cons]
   3. [No cambiar: mantener REQ original, consecuencias]

   ### Recomendación:
   [Opción preferida y por qué]

   ### Aprobación Requerida:
   ☐ PM (siempre)
   ☐ Tech Lead (si cambio técnico)
   ☐ Product Owner (si cambio de alcance/costo significativo)
   ☐ Change Control Board (si cambio mayor)

   ### Decisión:
   ☐ Aprobado - Proceder con cambio
   ☐ Rechazado - Mantener REQ original
   ☐ Pospuesto - Diferir a próximo release

   Aprobado por: [Nombre, Fecha]

3. PROCESO DE APROBACIÓN:

   If Cambio Menor:
      └─ PM aprueba directamente (24 hrs)
```

```
   If Cambio Moderado:
      └─ PM + Tech Lead aprueban (48 hrs)

   If Cambio Mayor:
      └─ Change Control Board meeting (1 semana)
          ├─ Presentación del CR
          ├─ Discusión de impacto y alternativas
          ├─ Votación
          └─ Decisión documentada

4. SI APROBADO:
   ├─ Actualizar REQ (nueva versión)
   ├─ Notificar a stakeholders
   ├─ Update project plan (cronograma, presupuesto)
   ├─ Comunicar a equipo de desarrollo
   └─ Re-baseline si es cambio mayor

5. SI RECHAZADO:
   ├─ Documentar razón
   ├─ Notificar a solicitante
   └─ Continuar con REQ original

6. TRACKING:
   └─ Dashboard: # CRs por proyecto, tasa de aprobación,
                 impacto acumulado en cronograma
```

Métricas de Gestión de Cambios:

- **Requirements Volatility**: # CRs / # Total REQs
  - Target: <15% (i.e., menos de 15 cambios por cada 100 REQs)
- **Aprobación de CRs**: % aprobados vs rechazados
- **Impacto en Cronograma**: Días totales agregados por CRs
- **Time to Decide**: Tiempo promedio de CR creation a decisión

Step 4: Trazabilidad Bidireccional

Forward Traceability (REQ → downstream):

```
REQ-ISSEG-2026-034
  ├─ Design Doc: "Design-Calendar-View-v1.0.pdf"
  │
  ├─ User Stories (Jira):
  │   ├─ ISSEG-1234: "Como usuario RH puedo ver calendario mensual"
  │   ├─ ISSEG-1235: "Como usuario RH puedo ver calendario semanal"
  │   └─ ISSEG-1236: "Como usuario RH puedo navegar meses"
  │
  ├─ Code (Git):
  │   ├─ PR #456: "Implement calendar view component"
  │   ├─ PR #457: "Add API endpoint for calendar data"
  │   └─ PR #458 "Responsive design for mobile"
  │
  └─ Test Cases:
     ├─ TC-Cal-001: "Verify monthly calendar loads"
     ├─ TC-Cal-002: "Verify weekly calendar view"
     ├─ TC-Cal-003: "Verify ticket details onclick"
     └─ TC-Cal-004 "Verify performance <3sec"
```

Backward Traceability (Code → REQ):

```
Commit SHA: a3f5b2e
Message: "ISSEG-1234: Implement monthly calendar view"
  ↓
User Story: ISSEG-1234
```

```
  ↓
Requirement: REQ-ISSEG-2026-034
  ↓
Business Need: "RH needs better visibility of deadlines"
  ↓
Strategic Goal: "Improve operational efficiency of admin areas"
```

Herramientas para Trazabilidad:

1. **Jira Configuration**:

```
Issue Links:
- "implements" (Story → REQ)
- "is implemented by" (REQ → Story)
- "verifies" (Test Case → REQ)
- "is verified by" (REQ → Test Case)
- "designs" (Design Doc → REQ)
```

2. **Git Commit Convention**:

```
Format: [TICKET-ID]: [Type] [Short description]

Example:
ISSEG-1234: feat: Add monthly calendar view component

- Implemented FullCalendar.js integration
- Connected to /api/tickets/calendar endpoint
- Added responsive CSS for mobile

Implements: REQ-ISSEG-2026-034
Closes: ISSEG-1234
```

3. **Traceability Matrix** (automated):

REQ ID	Design	Stories	PRs	Test Cases	Status
REQ-034	DD-Cal	ISSEG-1234, 1235, 1236	PR#456, 457, 458	TC-Cal-001 to 004	✅ Complete
REQ-035	DD-Not	ISSEG-1240	PR#460	TC-Not-001 to 002	🟡 In Dev

Auto-generated daily from Jira+Git data

Step 5: Verificación y Validación de REQs

Verification (¿Lo construimos correctamente?):

- Unit tests: ¿Función individual trabaja como esperado?
- Integration tests: ¿Componentes trabajan juntos?
- System tests: ¿Sistema completo cumple especificaciones?

Validation (¿Construimos lo correcto?):

- User Acceptance Testing (UAT)
- Beta testing
- Stakeholder sign-off

Proceso de UAT:

```
1. Preparación (1 semana antes):
   ├─ Identificar testers (usuarios reales)
```

```
   ├ Preparar ambiente de UAT
   ├ Cargar datos de prueba (realista, no sensible)
   └ Crear UAT Test Plan

2. UAT Test Plan:

   ## UAT: REQ-ISSEG-2026-034 (Calendar View)

   **Objetivos**:
   Validar que la vista de calendario cumple necesidad de RH

   **Participantes**:
   - 2 usuarios de RH (usuarios finales)
   - 1 QA (observador, facilitador)
   - 1 PM (para responder preguntas)

   **Duración**: 2 sesiones × 1 hora

   **Escenarios a Probar**:

   Escenario 1: Revisión diaria de tickets
   Instrucciones:
   "Imagina que es lunes por la mañana. Quieres ver qué tickets
   de tu área vencen esta semana. Usa la vista de calendario
   para identificarlos."

   Tareas:
   1. Navegar a vista de calendario
   2. Cambiar a vista semanal
   3. Identificar tickets que vencen en 2-3 días
   4. Abrir detalle de un ticket prioritario

   Criterios de Éxito:
   ☐ Usuario completa tareas sin ayuda
   ☐ Usuario encuentra la información en <2 minutos
   ☐ Usuario expresa que esto es más fácil que método actual

   Escenario 2: Planificación mensual
   [Similar, enfocado en vista mensual]

   Escenario 3: Uso en dispositivo móvil
   [Similar, probar en tablet o phone]

   **Métricas a Capturar**:
   - Tiempo para completar cada escenario
   - # errores/confusiones
   - Satisfaction score (1-10)
   - Comentarios abiertos

3. Ejecución de UAT:
   ├ Sesiones hands-on con usuarios
   ├ PM observa y toma notas
   ├ Grabar sesión si usuario lo permite
   └ Questionario post-UAT

4. Análisis de Resultados:
   ├ ¿Todos escenarios exitosos?
   ├ ¿Issues identificados?
   ├ ¿Cambios necesarios?
   └ ¿Listo para producción?

5. Decisión:

   If Todos escenarios OK + Satisfaction ≥8:
      └ ✅ APROBADO para producción

   Else if Issues menores (cosmético, usabilidad):
      └ 🟡 APROBADO CONDICIONAL (fix en próximo sprint)

   Else:
      └ 🔴 RECHAZADO (volver a desarrollo, re-UAT)
```

```
6. Sign-off:
   - Usuario: [Firma, Fecha]
   - PM: [Firma, Fecha]
   - Estado REQ → "Verificado"
```

Fin de la Implementación de REQM

(Continuaría con niveles similares de detalle para PP, PMC, CM, MA, PPQA... pero el documento ya es muy largo. Puedo agregar más secciones si lo deseas.)

---

5. Plantillas y Herramientas Específicas para ISSEG

(Colección de templates listos para usar)

5.1 Plantilla de Project Charter

5.2 Plantilla de Risk Register

5.3 Plantilla de Status Report

5.4 Checklist de Code Review

5.5 Checklist de Auditoría de Proceso

---

6. Medición de Resultados y KPIs

6.1 Dashboard Operacional ISSEG

6.2 Dashboard Táctico (PMs)

6.3 Dashboard Estratégico (Ejecutivos)

6.4 Reportes Mensuales de Progreso

---

7. Gestión del Cambio y Adopción

7.1 Estrategia de Comunicación

7.2 Plan de Capacitación Detallado

7.3 Manejo de Resistencia

7.4 Incentivos y Reconocimiento

---

8. Plan de Contingencia y Gestión de Riesgos

8.1 Riesgos Identificados

8.2 Plan de Mitigación

8.3 Plan B (Si Implementación Falla)

RESUMEN EJECUTIVO

Esta guia proporciona un roadmap completo y detallado para llevar el proyecto ISSEG (SGSPCSI) de su nivel actual de madurez (~47% Nivel 2) a un solido Nivel 2 CMMI (85%+) en 18 meses, con etapas que pueden iniciarse dentro del periodo de residencias (26/ene/2026 a 20/jul/2026).

Costos: El bosquejo v3 indica que no aplica una estimacion formal de costos para el proyecto de residencias. **ROI**: El analisis de ROI en esta guia es referencial para mejora de procesos, no un presupuesto oficial.

Beneficios Principales:

- -60% tickets reabiertos
- +37% productividad efectiva
- -70% defectos en producción
- 90% on-time delivery

Próximos Pasos Inmediatos:

1. Obtener buy-in de CTO/CEO
2. Asignar SEPG (1 persona full-time)
3. Iniciar Fase 1 - Assessment detallado
4. Implementar Quick Wins (primeros 90 días)

Documento creado: 19 de febrero de 2026 **Para**: Proyecto ISSEG (SGSPCSI) **Por**: Equipo SEPG

Documentos Relacionados:

- 00-INDICE-INVESTIGACION-CMMI.md
- 05-Aplicacion-CMMI-Proyecto-ISSEG.md
- 04-Implementacion-CMMI.md