

Contenidos

Prólogo	xix
Prefacio.....	xxi

Parte I. Tesis

1. ¿Qué es la ingeniería de <i>software</i> ?	1
Tiempo y cambio	4
Ley de Hyrum.....	6
Ejemplo: ordenación <i>hash</i>	7
¿Por qué no aspirar a que «nada cambie»?.....	8
Escala y eficiencia.....	10
Políticas que no escalan.....	11
Políticas que escalan adecuadamente.....	12
Ejemplo: actualización del compilador.....	13
Desplazamiento hacia la izquierda	15
Contrapartidas y costes	16
Ejemplo: rotuladores	18
Aportaciones a la toma de decisiones	19
Ejemplo: compilaciones distribuidas	19
Ejemplo: decidir entre tiempo y escala	21
Revisar decisiones, cometer errores	22
Ingeniería de <i>software</i> frente a programación	22
Conclusión.....	23
Resumen.....	23

Parte II. Cultura

2. Cómo trabajar bien en equipo	25
Ayúdeme a ocultar mi código	25
El mito del genio	26
La ocultación se considera perjudicial	28
Detección temprana	29
El factor autobús	29
Ritmo del progreso	30
En resumen, no se esconda.....	32
Todo es cuestión de equipo	32
Los tres pilares de la interacción social	33
¿Por qué importan estos pilares?	34
Humildad, respeto y confianza en la práctica	34
Cultura <i>post mortem</i> sin sentimiento de culpa	38
Ser Googley.....	40
Conclusión.....	41
Resumen.....	42
3. Compartir conocimientos	43
Desafíos para el aprendizaje	43
Filosofía.....	45
Preparación del escenario: seguridad psicológica	46
Tutoría.....	46
Seguridad psicológica en grupos grandes.....	47
Aumente sus conocimientos	48
Haga preguntas	48
Comprenda el contexto.....	49
Escalado de las preguntas: pregunte a la comunidad.....	50
Chats de grupo	50
Listas de correo electrónico	51
YAQS: plataforma de preguntas y respuestas	52
Escalado del conocimiento: siempre hay algo que enseñar	52
Horas de oficina	53
Charlas y clases de tecnología	53
Documentación.....	54
Código	56
Escalado de los conocimientos de la organización.....	57
Cultivar la cultura de compartir el conocimiento	57
Establecimiento de fuentes canónicas de información.....	59
Manténgase al día	62

Legibilidad: tutorías estandarizadas a través de la revisión del código	64
¿Qué es el proceso de legibilidad?	64
¿Por qué someterse a este proceso?	66
Conclusión.....	68
Resumen.....	69
4. Ingeniería para la equidad	71
Los prejuicios son la norma.....	72
Comprensión de la necesidad de la diversidad.....	74
Desarrollo de capacidades multiculturales.....	75
Hacer que se pueda procesar la diversidad	77
Rechazo de enfoques singulares.....	78
Desafío a los procesos establecidos	79
Valores frente a resultados	80
Mantener la curiosidad, seguir adelante.....	81
Conclusión.....	82
Resumen.....	82
5. Cómo liderar un equipo	83
Gerentes y líderes en tecnología (y ambos).....	83
El gerente de ingeniería.....	84
El líder en tecnología	84
El gerente líder de tecnología	84
Pasar de la función de colaborador individual a la función de liderazgo	86
Lo único que hay que temer es..., bueno, todo	86
Liderazgo de servicio	87
El gerente de ingeniería.....	88
«Gerente» es una palabra de cuatro letras	88
El gerente de ingeniería en la actualidad	89
Antipatronos.....	91
Antipatrón: contratar a personas fáciles de manejar	91
Antipatrón: ignorar a las personas de bajo rendimiento.....	92
Antipatrón: ignorar los problemas de carácter personal.....	93
Antipatrón: ser amigo de todos	94
Antipatrón: comprometer el listón de contratación	94
Antipatrón: tratar al equipo como si fueran niños.....	95
Patrones positivos.....	96
Perder el ego	96
Ser un maestro zen.....	97
Ser catalizador	99
Eliminar obstáculos	99

Ser maestro y mentor	100
Establecer metas claras	100
Ser honesto.....	101
Rastrear la satisfacción	103
La pregunta inesperada.....	103
Otros consejos y trucos	104
Las personas somos como las plantas	106
Motivación intrínseca frente a motivación extrínseca.....	107
Conclusión.....	109
Resumen.....	109
6. Liderazgo a escala	111
Siempre hay que decidir.....	111
La parábola del aeroplano.....	112
Identificación de las orejeras	113
Señalar las contrapartidas clave	113
Decidir y, después, repetir	114
Siempre hay que dejar solo al equipo.....	116
Su misión: formar a un equipo «autónomo»	117
División del espacio del problema.....	117
Siempre hay que mantenerse escalando	120
El ciclo del éxito	121
Lo importante frente a lo urgente.....	122
Aprender a dejar caer pelotas al suelo.....	124
Proteja su energía.....	125
Conclusión.....	127
Resumen.....	127
7. Medición de la productividad de la ingeniería	129
¿Por qué debemos medir la productividad de la ingeniería?	129
Triaje: ¿vale la pena medirlo?	131
Selección de métricas significativas con objetivos y señales	135
Objetivos	136
Señales	138
Métricas.....	139
Uso de datos para validar métricas.....	139
Actuar y realizar un seguimiento de los resultados.....	143
Conclusión.....	144
Resumen.....	144

Parte III. Procesos

8. Guías de estilo y normas.....	145
¿Por qué tenemos normas?.....	146
Creación de normas	147
Principios rectores	147
Guía de estilo	156
Cambio de las normas.....	159
El proceso.....	161
Árbitros de estilo.....	161
Excepciones	162
Orientación.....	163
Aplicación de las normas.....	164
Comprobadores de errores.....	166
Formateadores de código.....	167
Conclusión.....	169
Resumen.....	170
9. Revisión del código	171
Flujo de revisión del código.....	172
Cómo funciona la revisión de código en Google.....	173
Beneficios de la revisión de código.....	176
Corrección de código	177
Comprensión de código.....	179
Coherencia del código.....	180
Beneficios psicológicos y culturales.....	181
Compartir conocimientos	182
Mejores prácticas de la revisión de código	183
Sea cortés y profesional.....	183
Escriba cambios pequeños.....	184
Escriba descripciones de los cambios que tengan calidad.....	186
Mantenga al mínimo el número de revisores	186
Automatizar donde sea posible.....	187
Tipos de revisiones de código	187
Revisiones del código <i>greenfield</i>	188
Cambios de comportamiento, mejoras y optimizaciones	188
Corrección de errores y reversiones	189
Refactorizaciones y cambios a gran escala	190
Conclusión.....	190
Resumen.....	191

10. Documentación	193
¿Qué calificar como «documentación»?	193
¿Por qué es necesaria la documentación?	194
La documentación es como el código	196
Conozca a su audiencia	199
Tipos de audiencias	199
Tipos de documentación	201
Documentación de referencia	201
Documentos de diseño	204
Tutoriales	205
Documentación conceptual.....	207
Páginas de destino	208
Revisiones de la documentación.....	208
Filosofía de la documentación	210
QUIÉN, QUÉ, CUÁNDO, DÓNDE y POR QUÉ.....	211
El principio, la parte central y el final	212
Parámetros de la documentación de calidad.....	212
Documentación obsoleta	213
¿Cuándo necesita a escritores técnicos?.....	214
Conclusión.....	214
Resumen.....	215
11. Descripción general de las pruebas	217
¿Por qué escribimos pruebas?	218
La historia de Google Web Server	219
Pruebas al ritmo del desarrollo moderno	220
Escribir, ejecutar, reaccionar	222
Ventajas de probar el código	223
Diseño de un conjunto de pruebas.....	225
Tamaño de las pruebas.....	225
Alcance de las pruebas	230
Regla de Beyoncé	233
Nota sobre la cobertura del código.....	233
Pruebas a escala de Google	234
Dificultades de un gran conjunto de pruebas	235
Historial de pruebas en Google.....	237
Clases de orientación.....	238
Pruebas certificadas	239
Pruebas en los aseos.....	239
La cultura de pruebas actualmente	240
Límites de las pruebas automatizadas.....	241

Conclusión.....	242
Resumen.....	243
12. Pruebas unitarias	245
Importancia del mantenimiento.....	246
Prevención de pruebas frágiles	247
Esfuércese en lograr pruebas que no cambien	247
Pruebas a través de API públicas	249
Comprobar el estado, no las interacciones.....	252
Escriba pruebas claras	253
Haga que las pruebas sean completas y concisas.....	255
Pruebe los comportamientos, no los métodos.....	255
No ponga la lógica en las pruebas.....	261
Escriba mensajes de error claros.....	262
Pruebas y uso compartido de código: DAMP, no DRY	263
Valores compartidos	265
Configuración compartida	267
<i>Helpers</i> compartidos y validación.....	269
Definición de infraestructura de pruebas.....	269
Conclusión.....	270
Resumen.....	270
13. Dobles de pruebas	273
El impacto de los dobles de pruebas en el desarrollo del <i>software</i>	274
Dobles de pruebas en Google.....	275
Conceptos básicos.....	275
Un ejemplo de dobles de pruebas	275
Costuras.....	276
Marcos de trabajo de simulación.....	278
Técnicas para utilizar los dobles de pruebas	279
Simulación	279
<i>Stubbing</i>	279
Pruebas de interacción	280
Implementaciones reales	281
Preferencia de las implementaciones reales a las aisladas	281
Cómo decidir cuándo utilizar una implementación real	283
Simulación	285
¿Por qué son importantes las simulaciones?	286
¿Cuándo deben escribirse las simulaciones?	286
Fidelidad de las simulaciones.....	287
Las simulaciones se deben probar	288

¿Qué hacer si no hay una simulación disponible?	289
Stubbing.....	289
Los peligros de abusar del <i>stubbing</i>	290
¿Cuándo es adecuado utilizar <i>stubbing</i> ?	292
Pruebas de interacción	292
Preferencia de las pruebas de estado a las pruebas de interacción.....	292
¿Cuándo son apropiadas las pruebas de interacción?.....	294
Mejores prácticas para las pruebas de interacción	294
Conclusión.....	297
Resumen.....	297
14. Pruebas más grandes.....	299
¿Qué son las «pruebas más grandes»?.....	299
Fidelidad.....	300
Brechas frecuentes en las pruebas unitarias	301
¿Por qué no realizar pruebas más grandes?.....	303
Pruebas de gran tamaño en Google.....	304
Pruebas más grandes y el tiempo.....	305
Pruebas más grandes a escala de Google	306
Estructura de una prueba grande	308
El sistema bajo prueba.....	308
Datos para la prueba.....	313
Verificación	314
Tipos de pruebas más grandes	315
Prueba funcional de uno o más binarios interactivos.....	316
Pruebas de los navegadores y dispositivos	316
Pruebas de rendimiento, carga y estrés.....	316
Pruebas de la configuración de implementación	317
Pruebas exploratorias	318
Pruebas de regresión de diferencias A/B	319
UAT	320
Sistemas de sondeo y análisis canario	321
Recuperación en caso de catástrofe e ingeniería del caos.....	322
Evaluación al usuario	323
Grandes pruebas y el flujo de trabajo del desarrollador	324
Creación de pruebas grandes	325
Ejecución de pruebas grandes	325
Propiedad de las pruebas grandes	328
Conclusión.....	329
Resumen.....	330

15. Depreciación.....	331
¿Por qué hacer depreciación?.....	332
¿Por qué es tan difícil la depreciación?	333
Depreciación durante el diseño	335
Tipos de depreciación	336
Depreciación recomendada.....	336
Depreciación obligatoria.....	337
Advertencias de depreciación	339
Gestión del proceso de depreciación.....	340
Propietarios de los procesos	340
Hitos	341
Depreciación de las herramientas.....	342
Conclusión.....	343
Resumen.....	344

Parte IV. Herramientas

16. Control de versiones y gestión de ramas.....	345
¿Qué es el «control de versiones»?.....	345
¿Por qué es importante el control de versiones?.....	347
VCS centralizado frente a VCS distribuido.....	350
Fuente de verdad.....	353
Control de versiones frente a gestión de dependencias.....	355
Gestión de ramas	355
El trabajo en curso es similar a una rama.....	355
Ramas de desarrollo.....	356
Ramas de versión	358
Control de versiones en Google.....	359
One-Version	360
Escenario: varias versiones disponibles	361
La norma «One-Version»	362
Casi sin ramas longevas	362
¿Qué pasa con las ramas de versión?	364
Monorepos.....	365
Futuro del control de versiones	366
Conclusión.....	368
Resumen.....	369
17. Code Search	371
La interfaz de usuario de Code Search.....	372
¿Cómo utilizan los Googlers Code Search?.....	373

¿Dónde?.....	373
¿Qué?	374
¿Cómo?.....	374
¿Por qué?.....	374
¿Quién y cuándo?.....	375
¿Por qué una herramienta web independiente?	375
Escala	375
Vista global del código sin necesidad de configuración	376
Especialización	377
Integración con otras herramientas para desarrolladores.....	377
Presentación de las API.....	379
Impacto de la escala en el diseño	379
Latencia de la consulta de búsqueda	380
Latencia del índice	381
Implementación de Google	382
Índice de búsqueda	382
Clasificación	384
Selección de contrapartidas.....	387
Complejidad: repositorio en <i>head</i>	387
Complejidad: todos los resultados frente a los más relevantes	388
Complejidad: <i>head</i> versus ramas versus toda la historia versus espacios de trabajo	389
Expresividad: <i>token</i> frente a subcadena frente a expresión regular	390
Conclusión.....	391
Resumen.....	392
18. Sistemas de compilación y filosofía de la compilación	393
Propósito de un sistema de compilación.....	393
¿Qué sucede si no existe un sistema de compilación?	395
Pero todo lo que necesito ¡es un compilador!	395
¿ <i>Scripts</i> de <i>shell</i> al rescate?	396
Sistemas de compilación modernos.....	397
Todo se trata de dependencias	398
Sistemas de compilación basados en tareas.....	398
Sistemas de compilación basados en artefactos.....	403
Compilaciones distribuidas	410
Tiempo, escala, contrapartidas	414
Tratamiento de los módulos y las dependencias	415
La utilización de módulos detallados y la regla 1:1:1	415
Minimización de la visibilidad de los módulos	416
Gestión de dependencias	416

Conclusión.....	422
Resumen.....	423
19. Critique, herramienta de revisión del código de Google.....	425
Principios de las herramientas de revisión del código.....	425
Flujo de revisión de código.....	427
Notificaciones.....	428
Nivel 1: realización de un cambio.....	429
Diferenciaciones.....	429
Resultados del análisis.....	431
Integración estricta de herramientas.....	432
Nivel 2: revisión de la solicitud.....	433
Niveles 3 y 4: comprensión y comentarios del cambio.....	434
Comentarios.....	435
Comprensión del estado de un cambio.....	436
Nivel 5: aprobación del cambio (puntuación del cambio).....	438
Nivel 6: confirmación del cambio.....	440
Después de la confirmación: seguimiento del historial.....	440
Conclusión.....	441
Resumen.....	442
20. Análisis estático.....	443
Características del análisis estático eficaz.....	444
Escalabilidad.....	444
Usabilidad.....	444
Lecciones clave para hacer que el análisis estático funcione.....	445
Céntrese en la satisfacción de los desarrolladores.....	445
Haga que el análisis estático forme parte del flujo principal de trabajo del desarrollador.....	446
Permita la contribución de los usuarios.....	447
Tricorder: plataforma de análisis estático de Google.....	447
Herramientas integradas.....	449
Canales de retroalimentación integrados.....	450
Sugerencias de correcciones.....	451
Personalización por proyectos.....	451
<i>Presubmits</i>	453
Integración en el compilador.....	453
Análisis durante la edición y navegación por el código.....	454
Conclusión.....	455
Resumen.....	455

21. Gestión de dependencias	457
¿Por qué resulta tan difícil la gestión de dependencias?.....	459
Requisitos conflictivos y dependencias de diamante	459
Importación de dependencias.....	461
Promesas de compatibilidad	462
Consideraciones al hacer la importación.....	464
Cómo gestiona Google la importación de dependencias	465
Gestión de dependencias, en teoría.....	468
Nada cambia (también conocido como el «modelo de dependencias estáticas»)	468
Versionado semántico.....	469
Modelos de distribución por paquetes.....	471
Live at Head.....	471
Limitaciones de SemVer	473
SemVer puede que asegure una compatibilidad superior a la real	474
SemVer podría exagerar.....	474
Motivaciones	476
Minimum Version Selection	477
Entonces, ¿funciona SemVer?.....	478
Gestión de dependencias con recursos infinitos	479
Exportación de dependencias	482
Conclusión.....	486
Resumen.....	486
 22. Cambios a gran escala	 489
¿Qué es un «cambio a gran escala»?	490
¿Quién negocia con las LSC?.....	491
Obstáculos a los cambios atómicos	493
Limitaciones técnicas	493
Fusión de conflictos.....	494
Sin cementerios encantados	494
Heterogeneidad	495
Pruebas	496
Revisión de código.....	498
Infraestructura LSC	499
Políticas y cultura.....	500
Comprensión de la base de código	501
Gestión de cambios.....	502
Pruebas	502
Soporte del lenguaje.....	502
El proceso LSC	504

Autorización	504
Creación de cambios	505
Fragmentación y envío.....	506
Limpieza.....	509
Conclusión.....	510
Resumen.....	510
23. Integración continua	511
Conceptos de IC.....	513
Bucles de retroalimentación rápida.....	513
Automatización.....	515
Prueba continua.....	517
Desafíos de la IC.....	523
Pruebas herméticas.....	524
Integración continua en Google	527
Caso de estudio de IC: Google Takeout.....	530
Pero no puedo permitirme IC.....	537
Conclusión.....	537
Resumen.....	538
24. Entrega continua	539
Modismos de entrega continua en Google.....	540
La rapidez es un deporte de equipo: cómo dividir una implementación en partes manejables	541
Evaluación de cambios en el aislamiento: funciones de protección de banderas	542
La lucha por conseguir agilidad: configuración de un tren de lanzamiento.....	543
Ningún binario es perfecto	544
Cumpla con su fecha límite de lanzamiento	545
Calidad y enfoque en el usuario: envíe solo lo que se utilice	545
Desplazamiento a la izquierda: tomar antes decisiones basadas en los datos	546
Cambio de la cultura del equipo: creación de disciplina en el despliegue	548
Conclusión.....	549
Resumen.....	550
25. La computación como servicio	551
Dominio del entorno informático.....	552
Automatización del trabajo	552
Contenerización y tenencia múltiple	554
Resumen.....	557
Escritura de <i>software</i> para la computación gestionada.....	557

Arquitectura para el fracaso	557
Trabajos por lotes frente a trabajos de servicio.....	559
Gestión del estado.....	561
Conexión a un servicio.....	563
Código único	564
CaaS con el tiempo y la escala.....	565
Los contenedores como abstracción	565
Un servicio para gobernarlos a todos.....	568
Configuración enviada.....	570
Elección de un servicio informático.....	571
Centralización frente a personalización	573
Nivel de abstracción: sin servidores	575
Público versus privado	579
Conclusión.....	581
Resumen.....	582

Parte V. Conclusión

Epílogo.....	583
Índice.....	585