

Versionamento de Artefatos



Aqui voce encontra

Regras essenciais sobre versionamento de artefaos entregaveis de projeto, tais como pacotes para download.



Veja tambem

[Homologacao de Artefatos](#) para informacoes sobre o fluxo de evolucao de artefatos versionados do desenvolvimento ate a producao.

1. Visao Geral

O versionamento de artefatos eh a chave para controle e rastreabilidade dos entregaveis no projeto.

O objetivo eh localizar univocamente o artefato, e com isso ter informacoes sobre quais os elementos que o compoem e qual distribuicao do projeto ele esta atrelado.

2. Numeracao

O Processo 3PUP usa as regras da Fundacao Eclipse (1, 2) para versionamento e o novo padrao Java 6 (3) para depreciacao de artefatos entregaveis de codigo-fonte.

Enquanto o padrao Eclipse determina a profundidade dos niveis de numeracao e as regras essenciais para incremento de cada nivel, o padrao Java 6 delinea as regras para ruptura de API entre *major versions*.

Para um entendimento detalhado dessas regras, consulte os links de referencia ao final da pagina.

Para uma visao sintetizada dessas informacoes, use a tabela abaixo:

Nivel	Nome	Descricao
N.x.x.x	Nivel Maior	Indica uma versao de ruptura (ou possivel ruptura) de API, iniciando em 0 e incrementando em uma unidade cada vez manualmente pelo Gerente de Configuracao. A primeira versao de producao deve ter numero 1.
x.N.x.x	Nivel Menor	Indica alteracoes aditivas visiveis na API e que nao invalidem o nivel maior, inclusao de metodos novos metodos, criacao de metodos polimorficos, adicoes de parametros estilo "varargs", alteracoes de nomes de parametros, sobreescrita de tipos, alteracoes em hierarquia de classes ou interfaces e outras modificacoes que nao quebrem o Nivel Maior. Inicia em 0 e incrementa em uma unidade cada vez manualmente pelo Gerente de Configuracao. Quando o Nivel Mmaior muda de valor, o Nivel Menor reinicia em zero novamente.
x.x.N.x	Nivel de Servico	Indica correcoes de bugs ou melhorias nao visiveis em nivel de API, como adicao ou melhoria de documentacoes, tuning para performance, seguranca ou outras capacidades nao funcionais no sistema. Inicia em 0 e incrementa em uma unidade cada vez manualmente pelo Gerente de Configuracao. Quando o Nivel Menor reinicia, o Nivel de Servico reinicia em zero novamente.
x.x.x.N	Nivel de Qualificacao	Incrementado automaticamente a cada build, geralmente pelo sistema de integracao continua. Quando o Nivel de Servico reinicia, o Nivel de Qualificacao nao reinicia. DICA: Este Nivel de Qualificacao pode ser baseado em elementos como Timestamp, Numero do Build, Numero da Revisao do Repositorio, entre outras opcoes que identiquem claramente a unicidade dessa versao quando os valores de Nivel Maior, Menor e de Servico permanecerem iguais.

3. Exemplo Pratico

Data	Fase do Projeto	Alteracao de codigo	Versao de Desenvolvimento	Versao de Homologacao	Versao de Producao
15-Jan	0 - Iniciacao	Nenhuma	NA	NA	NA
23-Jan	1 - Anteprojeto	Primeiros commits de desenvolvimento de codigos com teste-base para validacao da arquitetura	0.0.0	NA	NA
12-Fev	1 - Anteprojeto	Mais commits de desenvolvimento para provas da arquitetura. Codigo quase estavel	0.0.0	NA	NA
17-Fev	1 - Anteprojeto	Projetista considerou arquitetura estavel o suficiente e vai iniciar homologacao dessa arquitetura com cliente. Ele faz o primeiro build de homologacao do projeto	0.0.0	1.0.0	NA

19-Fev	1 - Anteprojeto	Programadores continuam fazendo commits para ajustes e otimizações na arquitetura, seja qual for essa alteração	0.0.0	1.0.0	NA
21-Fev	1 - Anteprojeto	Durante homologação da arquitetura, projetista encontra um erro em uma de suas funcionalidades e puxa um dos commits de desenvolvimento para um novo build de correção na homologação	0.0.0	1.0.1	NA
23-Fev	1 - Anteprojeto	Mais um erro é encontrado na arquitetura, e exige outro ajuste, com um novo build de correção na homologação	0.0.0	1.0.2	NA
27-Fev	1 - Anteprojeto	As regras do jogo mudam, onde o cliente diz que precisa uma funcionalidade nova na arquitetura e não vai aceitar ela sem que isso seja implementado. A equipe de dev corre atrás disso e após vários commits de desenvolvimento, o projetista faz um novo build de homologação, agora com essa nova funcionalidade	0.0.0	1.1.0	NA
03-Mar	1 - Anteprojeto	Cliente homologa arquitetura Projetista faz primeiro build de produção do código da arquitetura, que vai ser usado como base para desenvolvimento do projeto (versão homologada vira versão de produção)	0.0.0	1.1.0	1.1.0
04-Mar	2 - Planejamento	Equipe antecipa alguns desenvolvimentos fazendo commits enquanto gestão do projeto faz estimativas e plano de entregas	0.0.0	1.1.0	1.1.0
15-Mar	2 - Planejamento	Equipe de gestão apresenta e homologa plano de entregas Enquanto isso equipe de dev continua seus desenvolvimentos internos para ganhar tempo	0.0.0	1.1.0	1.1.0
16-Mar	3 - Construção	Agora sim, projeto entra em fase de construção, com commits diários de desenvolvimento até ter funcionalidades estáveis	0.0.0	1.1.0	1.1.0
24-Mar	3 - Construção	Projetista elenca algumas funcionalidades como estáveis e faz primeiro build de homologação da fase de construção Projetista inicia homologação de funcionalidades Equipe dev continua desenvolvimentos	0.0.0	1.2.0	1.1.0
03-Abr	3 - Construção	Projetista elenca mais funcionalidades como estáveis e faz segundo build de homologação da fase de construção	0.0.0	1.3.0	1.1.0
04-Abr	3 - Construção	Equipe dev continua commits de desenvolvimento	0.0.0	1.3.0	1.1.0
05-Abr	3 - Construção	Cliente encontra erro em homologação e solicita ajustes imediatos para aprovação Equipe de dev diz que parte do código de desenvolvimento pode ser usado direto para este ajuste na homologação, sem necessidades de novos branches de trabalho Projetista eleva códigos de dev para homologação e faz novo build de homologação, a título de correções	0.0.0	1.3.1	1.1.0
06-Abr	3 - Construção	Cliente ainda não aceita homologação Projetista novamente coloca mais código de dev na homologação e gera novo build de ajuste na homologação	0.0.0	1.3.2	1.1.0
07-Abr	3 - Construção	Mais ajustes são necessários, e novamente projetista eleva código de dev para homologação e dispara novo build	0.0.0	1.3.3	1.1.0
08-Abr	3 - Construção	Dia anterior se repete, e exige outro build de ajuste na homologação. O quarto seguido sem nenhum incremento de funcionalidades.	0.0.0	1.3.4	1.1.0
09-Abr	3 - Construção	Cliente ainda não homologa ajustes realizados e projetista precisa fazer nova elevação de código dev para homologar Porém, equipe agora está com código muito diferente (já passaram 4 dias desde o primeiro dia de homologação) e agora exige tomada de decisão do projetista. Ou ele deriva inicia commits em homologação direto para correção direta na linha de homologação (o que vai gerar merges para o código dev que está mais atualizado) Ou ele discute com equipe dev as novas features do sistema não quebram as features antigas. Após discussão, projetista decide elevar todo código de dev para homologação, com novas features e ajustes funcionando, para isso ele gera um novo build de homologação, que acumula novas features e correções que já haviam sido feitas em relação a homologação atual	0.0.0	1.4.0	1.1.0
15-Abr	3 - Construção	Equipe dev continua commits em desenvolvimento Cliente homologa primeira sequência de features aptas para produção Projetista gera primeiro build de produção com features do projeto	0.0.0	1.4.0	1.4.0
12-Mai	3 - Construção	Quase 1 mês passou, e muitas features novas foram feitas em dev. Projetista eleva elas para nova homologação.	0.0.0	1.5.0	1.4.0
30-Mai	3 - Construção	Mais features sendo entregues pelo dev estão prontas para iniciar homologação Cliente ainda não homologou última versão do projeto	0.0.0	1.5.0	1.4.0
02-Jun	3 - Construção	Mais features sendo entregues pelo dev estão prontas para iniciar homologação, mas projetista ainda não as eleva Cliente homologa features e em consequência disso, projetista gera novo build de produção	0.0.0	1.5.0	1.5.0
03-Jun	3 - Construção	Projetista eleva feature de dev para iniciar nova homologação	0.0.0	1.6.0	1.5.0

04-Jun	3 - Construcao	Cliente inicia homologacao e encontra problemas Projetista faz ajuste direto no branche de homologacao mesmo para otimizar tempo, gerando novo build de homologacao Projegista faz merge do codigo ajustado da homologacao de volta para codigo de dev	0.0.0	1.6.1	1.5.0
05-Jun	3 - Construcao	Cliente aceita homologacao Projetista gera novo build de producao	0.0.0	1.6.1	1.6.1
07-Jun	3 - Construcao	Equipe dev inicia refatoracao completa do projeto, pois nova biblioteca esta sendo inserida na arquitetura e isso vai quebrar compatibilidade da versao atual Enquanto isso novas features tambem sao desenvolvidas	0.0.0	1.6.1	1.6.1
20-Jun	3 - Construcao	Equipe finaliza ajuste de refatoracao, e agora sistema tem tando features novas, quanto, principalmente, uma nova arquitetura incompativel com versao em uso na producao Projetista faz testes internos e considera validado iniciar uma nova homologacao, fazendo entao um build de homologacao para esta nova plataforma de codigo	0.0.0	2.0.0	1.6.1
22-Jun	3 - Construcao	Cliente inicia testes de novas features e arquitetura otimizada Equipe segue desenvolvendo na nova arquitetura features novas Codigo em producao continua com arquitetura antiga	0.0.0	2.0.0	1.6.1
24-Jun	3 - Construcao	Cliente homologa nova arquitetura Projetista faz build de producao	0.0.0	2.0.0	2.0.0
25-Jun	3 - Construcao	Cliente encontra erro na nova arquitetura Equipe dev corre atras para correcao usando o proprio trunk do projeto, corrige o problema e projetista eleva codigo e dispara novo build de homologacao desse ajuste	0.0.0	2.0.1	2.0.0
25-Jun	3 - Construcao	Cliente homologa ajustes da producao No mesmo dia, ao final da tarde, projetista faz build de correcao para producao	0.0.0	2.0.1	2.0.1
15-Jul	3 - Contrucao	Todas features do sistema sao finalizadas pela equipe dev Projetista as eleva e inicia ultima homologacao do sistema	0.0.0	2.1.0	2.0.1
21-Jul	3 - Construcao	Cliente homologa ultimas features Projegista gera build de producao com novas features	0.0.0	2.1.0	2.1.0
30-Jul	3 - Construcao	Cliente da aceite final no projeto e considera-o finalizado, sem mais coisas serem construidas Fase de construcao se encerra	0.0.0	2.1.0	2.1.0
31-Jul	4 - Repasse	Fase de repasse se inicia	0.0.0	2.1.0	2.1.0
01-Set	4 - Repasse	Cliente encontra erro do sistema Projetista aciona equipe, que utiliza ultimo codigo de dev (que esta sincronizado com a producao, pois quando ela entrou no ar nao havia ocorrido mais commits de dev) para montar uma homologacao de ajuste	0.0.0	2.1.1	2.1.0
02-Set	4 - Repasse	Cliente inicia testes para homologar ajustes propostos para producao	0.0.0	2.1.1	2.1.0
03-Set	4 - Repasse	Cliente aceita ajustes Projegista gera novo build de producao para publicar os ajustes homologados	0.0.0	2.1.1	2.1.1
04-Set	4 - Repasse	Cliente aceita todo o projeto, documentacoes, etc. Fase de repasse eh finalizada	0.0.0	2.1.1	2.1.1
05-Set	5 - Encerram ento	Inicia-se a fase de encerramento, visando garantia do projeto	0.0.0	2.1.1	2.1.1
07-Dez	5 - Encerram ento	Aos dois meses de garantia, cliente encontra erro grave Projetista aciona dev para correcao	0.0.0	2.1.1	2.1.1
09-Dez	5 - Encerame nto	Apos varios commits em dev, equipe tem codigo estavel para homologacao Projetista eleva codigo e dispara build de homologacao	0.0.0	2.1.2	2.1.1
10-Dez	5 - Encerram ento	Cliente inicia testes na homologacao	0.0.0	2.1.2	2.1.1
12-Dez	5 - Encerram ento	Cliente aceita homologacao Projetista eleva homologacao e dispara build de producao	0.0.0	2.1.2	2.1.2
05-Jan	5 - Encerram ento	Apos um ano de projeto e terminada garantia, projeto encerra-se	0.0.0	2.1.2	2.1.2

4. Referencias

1. <http://www.eclipse.org/equinox/documents/plugin-versioning.html>
2. http://wiki.eclipse.org/index.php/Version_Numbering

3. TODO: Link para padrao Java 6 de "deprecacao" de APIs