

Formaler Entwicklungsprozess elego

Nachfolgend ist der Softwareentwicklungsprozess beschrieben, so wie er bei elego ab dem nächsten DCCS-Release (1.1) gelebt werden soll.

Dieser Prozeß ist als verbindliche Vorgabe zu verstehen, die von allen Projektbeteiligten befolgt wird. Sie soll einen reibungslosen Projektablauf unter Einsatz der neuen Möglichkeiten von DITS (verteiltes ISSUE-Tracking) und DCVS (verteiltes CM mit Snapshot- und Changeset-Funktionalitäten) ermöglichen.

Ziel ist die Sicherstellung eines in sich geschlossenen konsistenten Entwicklungs- und Änderungsprozess zu gewährleisten, von der Anforderung bis zur Lieferung und dem Support.

Bitte Probleme bei der praktischen Arbeit mit dem Prozessmodell einfach mitteilen, so dass bei Schwierigkeiten bzw. Reibungen Anpassungen am Modell vorgenommen werden können.

Der Sprachgebrauch ist bereits an den des Vorgehensmodell CMII (Configuration Management II) angelehnt, welches das Configuration Management in den Mittelpunkt des Softwareentwicklungsprozesses stellt.

1 Behandlung von Anforderungen

1.1 Basics

Anforderungen sind alle zu behebenden Bugs und Feature Requests, also grundsätzlich alles was potentiell zu Änderungen (Changes) an der Software führen kann.

Anforderungen sind in sinnvolle nicht weiter teilbare Einzelaufgaben zu splitten. Im elego-Sprachgebrauch ist von **ISSUES** die Rede, die in Form von so genannten **ECRs** (Enterprise Change Requests) festgehalten werden.

Es ist unbedingt zu jeder Anforderung an die Software ein ECR zu erstellen. Ein ECR wird über die ISSUE-Trackingsoftware DITS erfaßt. Dabei wird jedem ECR eine eindeutige ID zugewiesen, so dass im Verlauf der weiteren Bearbeitung über die ID immer eindeutig Bezug auf den jeweiligen ECR genommen werden kann.

Alle Artefakte, die zu einem ECR gehören (Konzepte, Notizen, Ideen, Aufwände, Sourcecodeteile, beteiligte Personen, Testunterlagen, etc.) haben einen Bezug zum ECR über die ID bzw. über den Eintrag in der ISSUE-Tracking-Datenbank. Die ISSUE-Trackingsoftware ist das zentrale Kommunikationsinstrument im Prozeß

1.2 ECR-Attribute

Im folgenden sind die Attribute eines ISSUE beschrieben. Solange DITS noch nicht für den Echtbetrieb zur Verfügung steht, erfolgt die Steuerung der Entwicklung über Gnats. Ein Großteil der Attribute wird im DITS wiederzufinden sein. Teilweise müssen jetzt in Gnats „Krücken“ für Projektplanungsinformationen vereinbart werden, die später in DITS von Beginn an ordnungsgemäß verwaltet und dargestellt werden können.

Prio1

Synopsis (Subject) eine aussagekräftige Kurzbeschreibung des Problems bzw. der Erweiterung, damit klar wird worum es geht

Project Name des Projekts bzw. des Produktes

Category	Der Teilbereich der Software, für den der ISSUE relevant ist, folgende Bereiche für DCVS sind derzeit definiert,
	Softwareentwicklung an der Codebasis CVS: dcvs-cvs
	Softwareentwicklung an der Codebasis CVSup: dcvs-cvsup
	Softwareentwicklung an DCVS-Setupprozeduren: dcvs-setup

DCVS-Anwender-Dokumentation:	dcvs-doc
DCVS-Testprozeduren:	dcvs-test
Verschiedenes:	dcvs-misc

Class	Angabe der Art des ISSUE, Bug: PR Feature: CR
Confidential	Angabe ob der ISSUE für den Kunden/ die Öffentlichkeit sichtbar sein soll, sichtbar: Y nicht sichtbar: N
Creator	Name bzw. Kürzel desjenigen, der den ISSUE anlegt
CreationDate	Datum der Anlage des ISSUE
Status	aktueller Status des ISSUE (detaillierte Erläuterung in einem separaten Abschnitt)
Priority	Priorität in der Abarbeitung, hoch: high mittel: medium niedrig: low
Severity	Schweregrad (nur bei Bugs), hochkritisch (System läuft nicht): critical System läuft, aber erhebliche Beeinträchtigung: serious "Unschönheit", keine Systembeeinträchtigung: non-critical Features, Software-Änderungen sind immer: non-critical
Responsible	Name bzw. Kürzel desjenigen, der für die weitere Bearbeitung des ISSUE verantwortlich ist
Submitter	Name bzw. Kürzel desjenigen, der die letzte Änderung am ISSUE vorgenommen hat
Description	Ausführliche Beschreibung des ISSUE und seiner Umstände !Wichtig! Ein ISSUE ist so zu beschreiben, dass er auch für Dritte verständlich ist
Release	Release, in dem der Bug aufgetreten ist bzw. in welches ein CR integriert werden soll.

In Ermangelung eines separaten Feldes TargetRelease wird das TargetRelease mit einem „->“ angegeben. Bspw. ist ein Bug im Release 1.0.2 aufgetreten und soll in 1.0.3 behoben werden, so würde der Eintrag 1.0.2 -> 1.0.3 lauten. Das TargetRelease wird vom Projektmanager bzw. in Abstimmung mit diesem vergeben. Tritt der Bug/Verbesserungswunsch während der Entwicklungsphase eines Releases auf, wir die zukünftige Release-Nr. mit einem kleinen d davor angeben. Z. B. während der Entwicklung von 1.0.3:

d1.0.3 -> 1.0.3

Ein nach der Analyse/ Konzeption für den ISSUE geschätzter Aufwand ist ebenfalls in Ermangelung eines separaten Feldes hier einzutragen (Angabe in runden Klammern und mit Einheit:

Stunden = h)

Fix	In Ermangelung eines separaten Feldes hier Eintrag des Change Set Namens (1:1-Zuordnung ISSUE:Change Set) in eckigen Klammern + Beschreibung der Fehlerbehebung (nur bei Bugs), d.h. Beschreibung der Fehlerursache und was zur Behebung getan wurde
Releasenote	Information für Anwender zum ISSUE, die in den Releasenotes auftaucht
HowToRepeat	Sequence von Aktionen, die das Reproduzieren eines Fehlers ermöglichen (nur bei Bugs)

1.3 Umsetzungsplanung von ECRs

ECRs werden bewertet und für ein bestimmtes Release geplant. Eine definierte Zusammenstellung von ECRs bildet einen Änderungsauftrag zum bestehenden Release (Update) bzw. den initialen Auftrag. Im CMII-Sprachgebrauch ist für einen Änderungsauftrag, bestehend aus aufwandsmäßig bewerteten und eingeplanten ECRs, von **ECN** (Enterprise Change Notice) die Rede.

Todo: Hier ist zu beschreiben wie die Bewertung und Planung im Detail abläuft (Einfachverfahren und Verfahren über Change Review Board (CRB)) !Wann und in welchem Umfang ist ein Konzept zu erstellen? OW: Pragmatik -> Semantik -> Syntax: wann ist welches Feld dafür im ISSUE auszufüllen.

1.4 Lebenszyklus eines ECR

Ein ISSUE geht von Anlage bis zur Erledigung innerhalb des Teams durch verschiedene Hände und kann diverse Zustände annehmen. Insbesondere ist es wichtig vor Erledigung eine saubere Analyse, Konzeptionierung und Schätzung durchzuführen und erst nach Freigabe loszuarbeiten.

Die folgenden Status sind zunächst die verfügbaren in Gnats, die mit für den elego-Prozess sinnvollen Definitionen beschrieben sind und für einen reibungslosen Durchlauf von ISSUES so verwendet werden sollen.

open	der ISSUE ist neu angelegt worden und ist unbearbeitet durch Kunden, Entwickler, Projekt- oder Releasemanager
assigned*	ein ISSUE ist einem Verantwortlichen zugewiesen durch Projektmanager
analyzed	der ISSUE ist analysiert und eine Lösung konzeptionell erarbeitet worden, eine Aufwandschätzung liegt vor durch Entwickler
accepted*	der ISSUE ist zur Umsetzung freigegeben durch Projektmanager, bei einem Aufwand bis zu 4 h kann der Entwickler den PR auch selbst auf accepted setzen
rejected*	der ISSUE ist abgelehnt, eine Umsetzung erfolgt nicht

suspended*	die Erledigung des ISSUE ist aufgeschoben
feedback	der Entwickler hat den ISSUE bearbeitet inkl. Unit-Tests und gibt ihn für Integrations- und Systemtests frei durch Entwickler
closed	das Funktionieren des ISSUE ist während der nachgelagerten Tests überprüft und als gelöst begutachtet worden
reopened	ein <i>reopened</i> gibt es für den fehlerhaften Testfall unter Gnats derzeit nicht, somit ist der ISSUE wieder auf open zu setzen.
invalid	beim Versuch einen als Bug gemeldeten ISSUE zu reproduzieren wurde festgestellt, dass es sich nicht um einen Fehler sondern ein Missverständnis des Creators handelt -> in Ermangelung eines separaten Feldes wird der ISSUE auf <i>closed</i> gesetzt und in der Synopsis als (<i>invalid</i>) markiert
duplicate	es wird festgestellt, dass der ISSUE bereits in der Tracking-DB enthalten ist. Im Beschreibungsfeld Description ist auf den bereits vorhandenen gleichen Ersteintrag zu verweisen -> in Ermangelung eines separaten Feldes wird der ISSUE auf <i>closed</i> gesetzt und in der Synopsis als (<i>duplicate</i>) markiert

Das Setzen der einzelnen Status erfolgt durch den jeweiligen Verantwortlichen, wobei die mit „*“ gekennzeichneten Status in erster Linie vom Projektmanager gesetzt werden.

Bei Zustands- bzw. Verantwortungsübergang sind in den Kommentarfeldern aussagekräftige Begründungen (*Reason Changed*) einzugeben, so daß Begründungen für die entsprechenden Übergänge eingebbar sind und nachträglich mit Hilfe einer History-Funktion der gesamte Lebenszyklus nachvollzogen werden kann.

Wird ein PR nachträglich gesplittet, ist in die Synopsis ein Zusatz „splittet“ einzutragen und der PR auf closed zu setzen. Im audit trail (Log-Message) sind die PRs, die aus diesem PR entstehen und sie ersetzen, anzugeben sowie der Grund für das Splitting. In die neu entstehenden PRs ist in die Synopsis „descendant“ einzutragen und im audit trail (Log Message) der PR, von der er abstammt, und der Grund.

1.5 Statistik/ Status ECN

Todo: Hier ist zu beschreiben welche und wie Informationen aus der ISSUE-Trackingsoftware gezogen werden können, um einen Überblick über das Projekt (den Status der Bearbeitung der ECRs und die Aufwände) zu gewinnen.

2 Rollen im Entwicklungsprozeß

Elego arbeitet in der Regel pro Projekt mit einem Team bestehend aus einem oder mehreren Entwicklern und einem verantwortlichen Release Engineer, einem oder mehreren Testern und einem Gesamtverantwortlichen, dem Projektmanager.

Der Projektmanager ist verantwortlich für die Steuerung und Überwachung der einzelnen Prozessschritte.

Alle Rollen können je nach Art und Umfang des Projektes in Personalunion mit der Rolle als Entwickler ausgeführt werden.

3 Umsetzung von geplanten und konzeptionierten Anforderungen (Implementierung)

3.1 Startpunkt Baseline

Eine Baseline ist eine geschlossene Konfiguration, ein festgeschriebener Entwicklungszustand hoher Qualität auf die aufbauend eine geplante Entwicklung erfolgt.

3.2 Zielpunkt Baseline

Das Ziel der Implementierung ist die vollständige und fehlerfreie Implementierung aller eingeplanten ECRs auf dem kürzesten Weg, d.h. die möglichst effiziente Erstellung der geplanten Baseline.

3.3 Fixieren von Konfigurationen mit Snapshots

DCVS bietet gegenüber dem klassischen „Taggen“ von CVS die Möglichkeit, geschlossene Konfigurationen mit Hilfe von Snapshots festzuhalten. Dabei wird das Abbild der Konfiguration in einem unabhängigen Objekt fixiert. Die Entscheidung, wann Konfigurationen mit Hilfe dieser Funktionalität gekennzeichnet werden und die Anwendung dieser Funktion obliegt dem Release Engineer.

3.4 Entwicklungslinie

Die Entwicklung eines bestimmten Produktes bzw. Projektes kann als eine Abfolge von Baselines mit dazwischen liegenden Entwicklungsphasen, Tests und Qualitätsbewertungen entlang einer logischen Linie gesehen werden und soll als Entwicklungslinie bezeichnet werden.

3.5 Arbeit mit Change Sets

Die Implementierung eines ECR zieht die Änderung einer bestimmten Anzahl von Dateien nach sich. DCVS bietet nun die Möglichkeit, diese geänderten Dateien in Form eines Change Set festzuhalten. Dabei wird das Delta als Abbild von 2 Konfigurationen (Bezugskonfiguration -> neu erstellte Konfiguration) in einem unabhängigen Objekt fixiert. Die Anwendung dieser Funktionalität steht allen Entwicklern offen und ist konsequent zu verwenden, d.h. pro getestetem* ECR ist ein Changeset zu erstellen.

Es ist darauf zu achten, dass nicht Änderungen zu mehreren ECRs mit einem Commit eingecheckt werden, da diese sonst später nicht mehr getrennt behandelt werden können. Beim Commit ist ein aussagekräftiger Kommentar hinsichtlich der vorgenommenen Änderungen einzugeben, sowie jedesmal die betreffende ECR-ID.

*Hinweis: Mit Implementierung eines ECR ist seitens des Entwicklers stets ein Unit-Test durchzuführen, d.h. die durchgeführte Änderung ist hinsichtlich ihrer Funktion so genau wie möglich in der Entwicklungsumgebung zu testen! Jeder Fehler, der in nachfolgenden Tests bzgl. des ECR gefunden wird, verteuert die Entwicklung durch zusätzliche Kommunikation/ Dokumentation, neuen Release Candidate, Installation, etc. um ein Vielfaches!

Fertiggestellte Change Sets werden unter Kontrolle des Release Engineer in die aktuelle Entwicklungslinie integriert.

3.6 Erstellen eines Release Candidate

Sind alle für die Ziel-Baseline geplanten Anforderungen implementiert, per Unit-Test geprüft und entsprechende Change Sets erstellt worden, so wird seitens des Release Engineers der Entwicklungsschluß an die Entwickler kommuniziert.

Entwicklungsschluß bedeutet für die Entwickler, dass die Arbeiten in der aktuellen Entwicklungslinie eingestellt werden. Diesen Punkt der Entwicklung bezeichnen wir auch als Codefreeze.

Der Release Engineer checkt nun die aktuelle Entwicklungskonfiguration aus und integriert ggf. noch nicht enthaltene Change Sets. Nach Beendigung von definierten Tests in seiner Entwicklungsumgebung checkt er die Konfiguration ein, bei dieser Konfiguration sprechen wir vom

Release Candidate und sie wird mit Hilfe eines Snapshot als solcher fixiert.

Der Release Candidate wird nun durch Integrationstests nach einem definierten Testplan auf seine Qualität geprüft, die Testergebnisse werden über die ISSUE-Trackingsoftware den ECRs eindeutig zuordenbar dokumentiert (Details sind im Kapitel Softwaretests beschrieben -> Todo: Kapitel Test erstellen).

3.7 Festlegen einer Baseline

Integrations- und Abnahmetests führen je nach Qualität des Release Candidate zur Nachbearbeitung von ECRs bzw. zu neuen ECRs.

Oberstes Ziel ist es immer, Nachbearbeitungen zu vermeiden und gleich im ersten Ansatz einen qualitativ hochwertigen Release Candidate herzustellen, denn jede zusätzliche Iteration (Planung, Konzeption, Implementierung, Test, Kommunikation, etc.) kostet zusätzliches Geld.

Der Release Engineer legt nach den Integrations- und Abnahmetests durch die Bewertung der ggf. noch offenen ECRs und deren Severity fest, ob der betreffende Release Candidate stable, also einer Baseline würdig ist.

Im Falle des positiven Entscheids wird der Release Candidate zur Baseline erhoben.

3.8 Kennzeichnung von Release Candidates und Baselines

Snapshots für Release Candidates und Baselines werden in separaten Verzeichnissen gehalten, pro Entwicklungslinie sind dies:

Release Candidate: RC/

Baseline: BL/

Die Benennung der Release Candidates erfolgt nach dem Schema:

<NAME>_<MAJOR>_<MINOR>_<LFD.-NUMBER>

wobei <MAJOR>_<MINOR> die Nummer des geplanten Release und NUMBER eine eindeutige fortlaufende Nummer darstellt. Für NAME sollte der Projekt- oder Produktname eingesetzt werden.

Wird ein Release Candidate zur Baseline erhoben, so wird die zugehörige Datei einfach aus dem Verzeichnis RC/ nach BL/ kopiert und die lfd. Nummer entfernt.

3.9 Kennzeichnung von Change Sets

Change Sets werden pro Entwicklungslinie im Verzeichnis CS/ abgelegt, sie werden nach folgendem formalen Schema benannt:

<NAME>_<NUMBER>

wobei für NUMBER die ID des ECR mit PRÄFIX „ECR-“ eingesetzt werden sollte, also

Nummer = ECR-<ECR-ID>

Für NAME sollte der Projekt-, Produkt-, und/ oder Kundenname eingesetzt werden.

3.10 Branching innerhalb der Entwicklungslinie – Release Branch

Branchen zum Zweck der Weiterentwicklung unabhängig von ggf. noch anfallenden Nachbearbeitungsaktivitäten sollte man grundsätzlich nur ausgehend von einer stabilen Baseline. Die Entscheidung zur Erstellung eines Branch auf der Ebene logischer Entwicklungslinien fällt der Release Engineer in Abstimmung mit dem Projektmanager.

Sinnvollerweise ist mit Festlegung einer Baseline auch gleichzeitig ein logischer Release Branch anzulegen, der die definierte Problembehebung durch Anwendung von ECR bezogenen Change Sets gegen die festgelegte Baseline auf dem separaten Branch ermöglicht.

Die Anwendung von Change Sets auf dem Release Branch erfolgt durch den Release Engineer selbst oder auf dessen Anweisung.

Die Kennzeichnung des auf der Baseline basierendes Release Branches erfolgt nach dem Schema:

RELEASE_(BRANCH)_<BASELINE>_HEAD

Für BASELINE ist die Bezeichnung der zugrunde gelegten Baseline einzusetzen.

Für jede Änderung, für jeden Fix sind Fix- bzw. Change-Branches anzulegen. Die Bugfixes werden nachträglich sowohl in den Head des Release-Branches als auch in den Head des Trunks gemerged, bzw. die entsprechenden Change Sets werden angewendet. Für Bugfixes wird vom Head gebrancht anstatt von der Baseline. [FIXME: Vom Trunk oder vom Releasebranch? Wer wendet die Change Sets an? Ausschließlich der Release-Manager? Was ist mit den Zwischenstadien vor der nächsten Baseline?]

3.11 Verteiltes Arbeiten mit DCVS

Ein verteiltes Arbeiten unter DCVS unterscheidet sich aufgrund der Möglichkeiten der Nutzung von Snapshots und Change Sets nicht wesentlich von der zentralistischen Arbeitsweise.

Der einzige Unterschied besteht darin, dass Change Sets dezentral erstellt und durch den Release Engineer in einer Entwicklungslinie entsprechend 3.6 „Erstellung Release Candidate“ auf genau einem DCVS-Server (Release-Server) zusammengeführt werden.

4 Softwaretests

4.1 Unit-Tests

Todo: Hier ist zu beschreiben was Sinn und Zweck des Unit-Test sind, was er umfaßt, wie er möglichst effektiv durchzuführen und was zu dokumentieren ist

4.2 Integrationstests

Todo: Hier ist zu beschreiben was Sinn und Zweck des Integrations-Test ist, wie er vorbereitet werden muß (Testplanung, Testszenarien) und was zu dokumentieren ist (Abnahmedokumentation)

4.3 Regressionstests

Todo: Hier ist zu beschreiben was Regressionstests beinhalten und wie diese vorzubereitet, durchgeführt und protokolliert werden

Glossar

CMII	Configuration Management II
	ist ein Vorgehensmodell dessen Kern ein konsistentes Configuration Management aller Artefakte bildet und einen geschlossenen Änderungsprozess einschließt
CRB	Change Review Board
	besteht aus den Projekt- und kaufmännisch Verantwortlichen der Auftraggeber- und Auftragnehmerseite
CIB	Change Implementation Board
	setzt sich aus den entsprechend technisch Verantwortlichen zusammen
CS	Change Set
	Ergebnis einer DCVS-Funktion, die ein Entwicklungs-Delta in Bezug auf eine geschlossene Konfiguration in einem separaten Objekt fixiert. Das CS wird innerhalb des Objekts durch die geänderten Revisionsnummern der betroffenen Files repräsentiert.
ECR	Enterprise Change Request
	steht für jedwede Anforderung an das System, ob Fehlerbehebung oder funktionale Anforderung
ECN	Enterprise Change Notice
	ist eine Zusammenstellung von bewerteten und eingeplanten ECRs
SN	Snapshot
	Ergebnis einer DCVS-Funktion, die eine geschlossene Konfiguration in einem separaten Objekt fixiert. Der SN wird innerhalb des Objekts durch die Revisionsnummern der betroffenen Files repräsentiert.