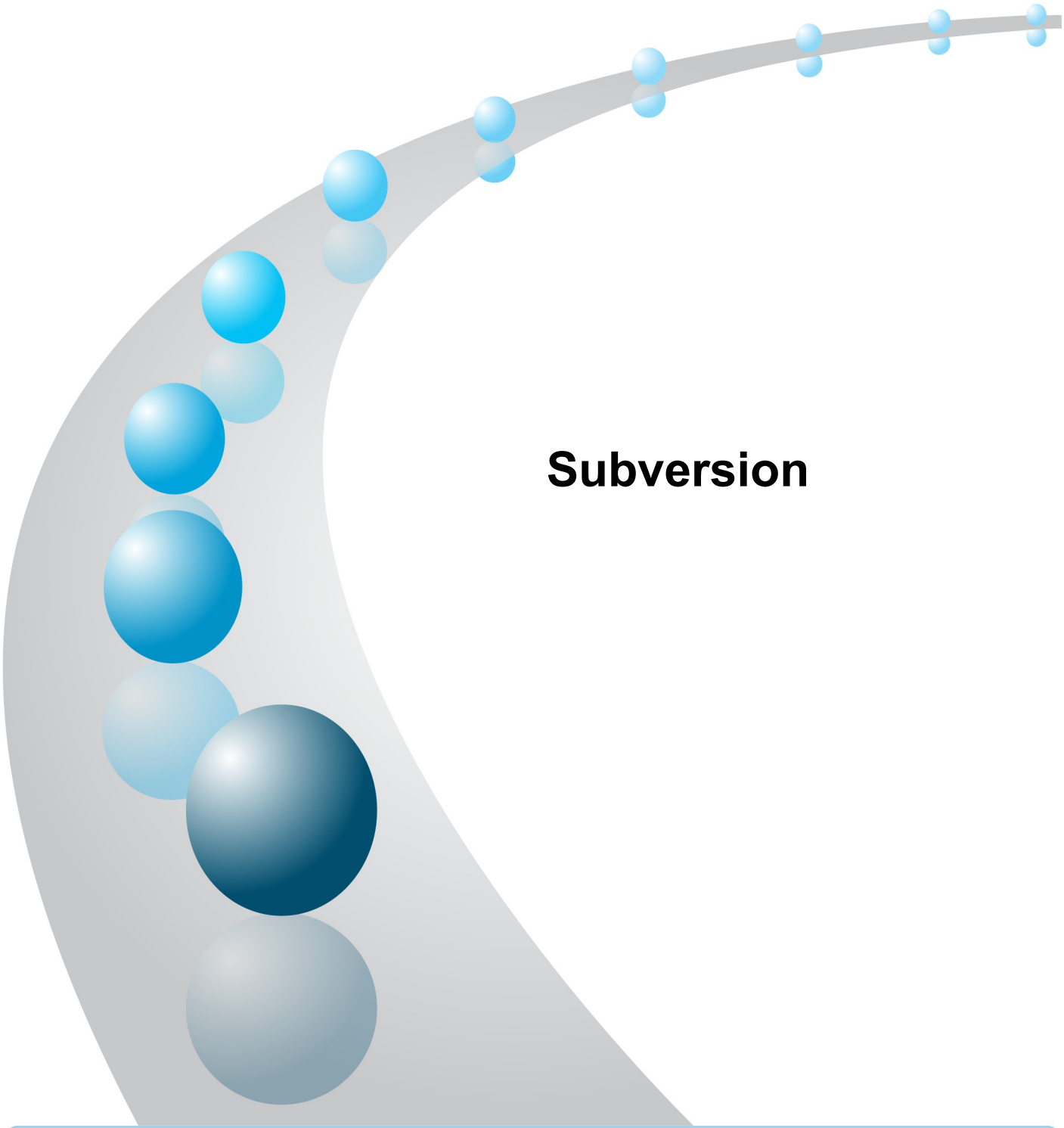
Noël Vaes

Java Trainer & Consultant



Subversion

Roode Roosstraat 5
3500 Hasselt
België

+32 474 38 23 94
noel@noelvaes.eu
www.noelvaes.eu

Vrijwel alle namen van software- en hardwareproducten die in deze cursus worden genoemd, zijn tegelijkertijd ook handelsmerken en dienen dienovereenkomstig te worden behandeld.

Correspondentie inzake overnemen of reproductie kunt u richten aan:

Noël Vaes
Roode Roosstraat 5
3500 Hasselt
België

Tel: +32 474 38 23 94

noel@noelvaes.eu
www.noelvaes.eu

Ondanks alle aan de samenstelling van deze tekst bestede zorg, kan de auteur geen aansprakelijkheid aanvaarden voor eventuele schade die zou kunnen voortvloeien uit enige fout, die in deze uitgave zou kunnen voorkomen.

06/07/2015

Inhoudsopgave

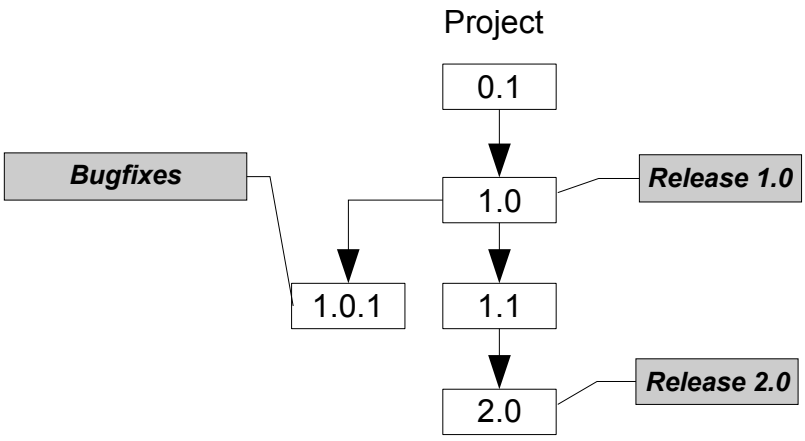
Hoofdstuk 1: Subversion

1.1 Inleiding

Bij grotere programmeerprojecten waarbij tevens meerdere personen betrokken zijn, krijgt men te maken met een aantal typische problemen.

Zo doorloopt de ontwikkeling van de software **verschillende stadia**: er ontstaan verschillende versies van de software waaraan telkens wordt verder gebouwd. Op een bepaald moment doet men een *release* en wordt de software vrijgegeven voor uitvoerige testen en uiteindelijk wordt de software geschikt bevonden voor de eindklant. Na deze eerste versie komen er vaak nog aanpassingen die uiteindelijk via de nodige tussenversies resulteren in een nieuwe officiële *release*.

Het eindproduct is de laatste versie in de rij van opeenvolgende versies waarbij telkens



aanpassingen gedaan werden aan de bestaande versie.

Daar komt nog bij dat vaak op basis van een bestaande versie **parallel** gewerkt wordt aan kleine aanpassingen aan deze versie (*bug fixes*) en aan de voorbereiding van een nieuwe grote release met allerlei nieuwe functionaliteiten. Sommige delen van de code zijn hetzelfde, andere delen worden in het nieuwe traject helemaal herwerkt terwijl de oorspronkelijke code nog wordt onderhouden voor de ondersteuning van de oudere versie.

Voorts wordt aan dergelijke grote programmeerprojecten meestal met **meerdere mensen** gewerkt. Samen delen ze dan dezelfde broncode waarbij ieder verantwoordelijk is voor een deel van die broncode. De broncode moet daarom centraal beheerd worden zodat iedere programmeur er toegang toe heeft, de code kan compileren en uitvoeren en er tenslotte zijn eigen wijzigingen in kan aanbrengen. De andere programmeurs moeten nadien deze wijzigingen kunnen overnemen.

Vaak gebeurt het dat twee programmeurs tegelijkertijd bezig zijn met hetzelfde stuk broncode. Dan is het belangrijk dat de wijzigingen die door beide programmeurs gedaan werden, op correcte wijze geïntegreerd worden.

Bij de ontwikkeling van *Open Source* projecten worden deze problemen nog erger. Hier wordt door heel veel programmeurs gewerkt aan eenzelfde project dat meestal ook nog allerlei stadia heeft met heel wat vertakkingen.

Bovenstaande problemen maken een systeem noodzakelijk dat de broncode van een programmeerproject centraal beheert en de verschillende versies van de software bijhoudt en toegankelijk maakt. Dit noemt men een systeem voor versiebeheer (*version control system*).

Momenteel zijn er verschillende systemen op de markt zoals *ClearCase, PVCS, SourceSafe* enz... Uit de *Open Source* wereld is inmiddels een *defacto* standaard gegroeid voor versiebeheer: ***Concurrent Versions System*** (CVS) en diens opvolger ***Subversion.*** Nagenoeg alle *Open Source* Java projecten maken inmiddels gebruik van een van deze versiebeheersystemen en ook voor interne projecten wordt het meer en meer gebruikt.
In deze cursus nemen we ***Subversion*** onder de loep. *Subversion* is een verbeterde variant van CVS die een aantal tekortkomingen van CVS oplost.

Subversion is een versiebeheersysteem dat bestanden en mappen ter beschikking stelt van verschillende gebruikers en dat tevens de geschiedenis hiervan bijhoudt. Alle wijzigingen van de bestanden en mappen worden bijgehouden zodat het steeds mogelijk is een bepaalde toestand terug te creëren. De centrale opslagplaats van de bestanden en mappen noemt men de *repository*.

Subversion is dus eigenlijk een soort bestandsserver die ook de geschiedenis van de bestanden en de mapstructuur bijhoudt. Hierdoor kan *Subversion* ook voor andere doeleinden gebruikt worden waar het bijhouden van de geschiedenis van bestanden belangrijk is. Dit is dus niet beperkt tot programmeerprojecten.

Alle informatie over *Subversion* is te vinden op volgende website:
http://subversion.apache.org/

1.2 Soorten versiebeheer

Om de werking van *Subversion* te begrijpen moeten we eerst de problematiek van gedeelde bestanden verder bekijken. Indien twee gebruikers van eenzelfde centraal bestand gebruik willen maken en dit bestand willen aanpassen, kunnen er conflicten optreden.

Gebruiker A kan een bestand uit de *repository* halen, dit aanpassen en het gewijzigde bestand terug wegschrijven in de *repository.*

Gebruiker B kan dat zelfde bestand ook lezen en aanpassen. Indien hij echter zijn wijzigingen iets later dan A wegschrijft, gaan de wijzigingen van A verloren.

Om dit probleem op te lossen zijn er twee mechanismen:

1.2.1 Lock-modify-unlock (pessimistic locking)

Een eerste mechanisme is het *lock-modify-unlock* mechanisme. Men noemt dit ook wel *pessimistic locking.* Hierbij zal gebruiker A het bestand lezen en onmiddellijk vergrendelen. Nadat hij wijzigingen heeft aangebracht kan hij het nieuwe bestand terug wegschrijven en wordt het bestand ontgrendeld. Zolang het bestand vergrendeld is, kan een andere gebruiker het bestand niet wijzigen. Alleen degene die het bestand vergrendeld heeft, kan wijzigingen aanbrengen.
Gebruiker B moet dus wachten totdat het bestand ontgrendeld is vooraleer hij een wijzigbare versie van het bestand kan bemachtigen.

Dit mechanisme heeft echter een aantal nadelen:

1. Indien gebruiker A **vergeet** het bestand terug te ontgrendelen kan gebruiker B niet verderwerken. De beheerder moet dan het bestand manueel terug ontgrendelen.

2. Gebruiker B moet altijd wachten totdat het bestand ontgrendeld is, ook al zijn de wijzingen die hij maakt **niet** noodzakelijk **conflicterend** met de wijzigingen die gebruiker A maakt. Denken we bijvoorbeeld aan een broncode-bestand waarbij de verschillende programmeurs op verschillende plaatsen wijzigingen aan het aanbrengen zijn, zonder dat die invloed hebben op elkaar. Er is dus heel wat vertraging bij het samenwerken aan eenzelfde bestand.
3. Het vergendelen van een bestand kan de **valse illusie** wekken dat er niets mis kan gaan. Het kan evenwel gebeuren dat een bestand afhankelijk is van een ander bestand dat niet vergrendeld is en dus wel gewijzigd kan worden door een andere gebruiker. Het uiteindelijke resultaat kan dan toch nog foutief zijn.

1.2.2 Copy-modify-merge (optimistic locking)

Een ander mechanisme is *copy-modify-merge* waarbij de gebruiker een lokale kopie neemt van het bestand en hierin aanpassingen doet. Nadien zal hij het gewijzigde bestand terug proberen te plaatsen in de centrale *repository*. Indien het centrale bestand intussen door iemand anders gewijzigd is, wordt getracht de wijzigingen te versmelten (*merge*). Indien dit niet lukt, moet het conflict opgelost worden door de gebruiker.

Dit systeem is veel flexibeler omdat hier geen bestanden vergrendeld worden en gebruikers tegelijkertijd wijzigingen kunnen aanbrengen. De wijzigingen worden indien nodig automatisch samengevoegd en enkel indien er een conflict optreedt, zal de gebruiker deze samenvoeging manueel moeten doen.

Men noemt dit ook wel *optimistic locking* omdat men er van uit gaat dat er geen conflicten zullen zijn (*optimistisch*) en als ze er toch zijn, moeten ze maar opgelost worden.

Subversion gebruikt het mechanisme van *copy-modify-merge* omdat hiermee de meest vlotte samenwerking tussen verschillende gebruikers gerealiseerd wordt. In sommige gevallen is vergendeling evenwel noodzakelijk en om die reden wordt ook dit mechanisme ondersteund.

1.3 De repository

Subversion houdt mapstructuren en hun bestanden bij in een centrale opslagplaats, de *repository* genoemd. Deze *repository* wordt doorgaans geplaatst op een centrale server en via het netwerk ter beschikking gesteld. *Subversion* is dus een *client-server* systeem.

De configuratie van de *repository* en de *server* valt buiten het bestek van deze cursus. We gaan gebruik maken van een bestaande *repository* die te vinden is op server ***noelvaes.eu***

Een *repository* kan benaderd worden via verschillende protocollen:

1. **http://hostname[:port]/*path/to/repository***
2. **https://hostname[:port]/*path/to/repository***
3. **svn://hostname[:port]/*path/to/repository***
4. **svn+ssh://hostname[:port]/*path/to/repository***
5. **file://*path/to/repository***

Het meest gebruikte protocol is HTTP waarbij heel de communicatie via een HTTP-tunnel verloopt. Dit veronderstelt evenwel het gebruik van een *Apache* module aan de kant van de server. Daarnaast is ook het specifieke *Subversion (svn)* protocol mogelijk dat gebruik maakt van een eigen poort (standaard 3690). Dit protocol is sneller en efficiënter maar veronderstelt wel dat eventuele firewalls dit doorlaten.
Het SVN-protocol kan ook getunneld worden over SSH zodat de gegevens versleuteld worden.

In deze cursus gebruiken we het *svn*-protocol. De repository is te vinden op volgende URL:

svn://noelvaes.eu/svn/student

login: student
wachtwoord: student123

Om de *repository* te benaderen, kunnen we gebruik maken van *commandline* applicaties. Deze kan men afhalen op de site http://subversion.tigris.org of http://subversion.apache.org . De meeste IDE's hebben echter een geïntegreerde *Subversion-client* die het werken met *Subversion* makkelijker maakt. Het is tevens mogelijk *Subversion* te integreren in het *Windows*-bestandssysteem met de applicatie ***TortoiseSVN*** die te vinden is op http://subversion.tigris.org

In deze cursus maken we gebruik van de *plugin* ***Subversive*** van *Eclipse*.

1.4 Subversion in Eclipse 3

Ook *Eclipse* heeft geen standaard integratie van *Subversion*, maar we kunnen deze *plugin* als volgt toevoegen:

We selecteren ***Help->Eclipse Marketplace.***
Zoek op ***Subversive*** en installeer ***Subversive SVN Team Provider***.
Selecteer alle onderdelen.

Bij het eerste gebruik van *Subversive* wordt gevraagd een *connector* te selecteren.
Selecteer de laatste versie van ***SVN Kit 1.x.y***

Installeer de connector en start nadien *Eclipse* opnieuw op.

1.5 Subversion gebruiken

Bij het gebruik van *Subversion* worden een aantal stappen doorlopen: