Strumenti di gestione del ciclo di vita del *software*

Università degli studi di Padova
a.a. 2008/09

Laurea in Informatica

Corso di Ingegneria del Software mod. A.

presenta

Nicola Bertazzo
nicola.bertazzo@gmail.com

Venerdì 17 ottobre 2008

> Sommario

1. Contesto e obbiettivi
2. Processo
3. Conclusioni
4. Esempi

> Sommario

1. Contesto e obbiettivi
2. Processo
3. Conclusioni
4. Esempi

Contesto e obbiettivi

> Chi sono io

- Neolaureato nella Laurea Specialistica in Informatica Università degli studi di Padova

- Allievo del Progetto Formativo Misura D4 dal titolo: “Studio e applicazione di metodologie e tecniche per misurare e migliorare la qualità del *software* ”
  - Approvato dalla regione Veneto e finanziato dal FSE
  - 1200 ore comprensive di 300 ore di *stage*
  - Argomento principale della tesi di laurea

Contesto e obbiettivi

> Ciclo di vita di prodotto software

Contesto e obbiettivi

> Ciclo di vita di prodotto software

Stabilire e seguire un **processo di sviluppo e verifica del software** ripetibile, con lo scopo di misurare e migliorare la qualità di prodotto

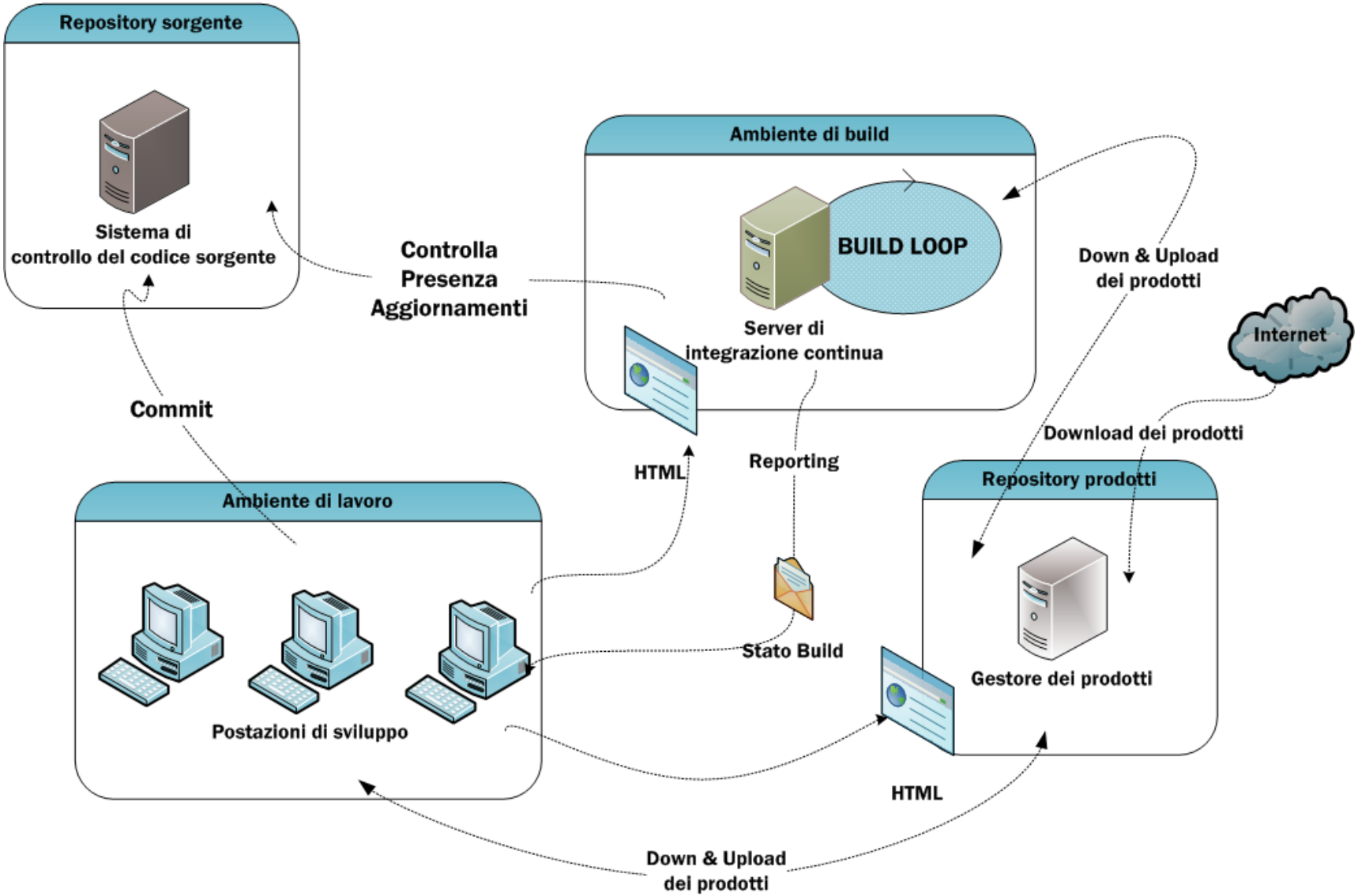
Applicare ***Best Practice*** e utilizzare **strumenti** per implementare un processo di sviluppo e verifica del software standard

> Sommario

1. Contesto e obbiettivi
2. Processo
3. Conclusioni
4. Esempi

Processo di sviluppo

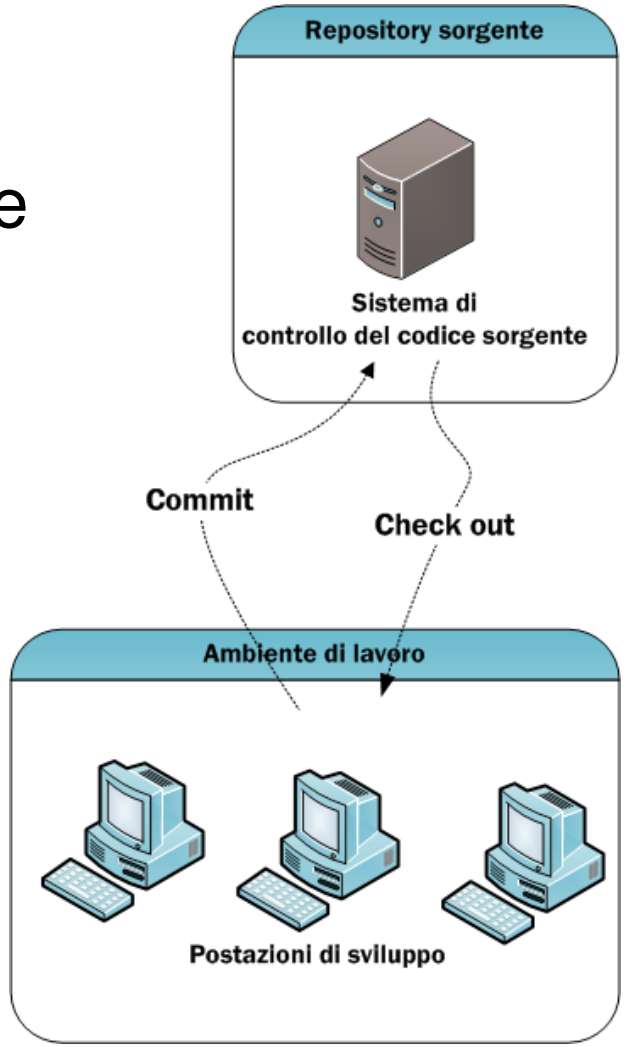
> Visione generale



Visione in dettaglio

> *Repository* Sorgente

- Deposito unico di tutto il codice sorgente
- Accesso alla storia completa del software
- Documentazione delle modifiche:
  - Autore
  - Oggetti modificati
  - Data e ora
  - Commenti e motivazioni
- Possibilità di gestire più rami diversi di sviluppo contemporaneamente



Repository Sorgente

> Strumenti

- CVS (www.nongnu.org/cvs)
- SVN (http://subversion.tigris.org)
- Tortoisecvs (http://www.tortoisecvs.org/)
- Tortoisesvn (http://tortoisesvn.tigris.org)
- Subclipse (http://subclipse.tigris.org)
- Subversive (www.eclipse.org/subversive)

Visione in dettaglio

> Ambiente di lavoro

Luogo dove avviene l'attività di sviluppo e verifica del prodotto

- Vengono realizzati i test di unità
- Il processo di costruzione del prodotto deve essere automatico
- Il codice prodotto viene inviato frequentemente al *Repository* dei sorgenti
- Dopo la realizzazione di una funzionalità avvengono le altre attività di verifica

Ambiente di lavoro

> Strumenti di verifica

Unità: Controllo del comportamento di ogni singolo oggetto in isolamento

- JUnit (http://www.junit.org/)
- JUnitPerf (http://clarkware.com/software/JUnitPerf.html)

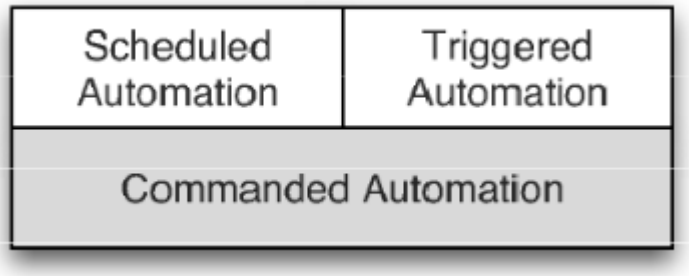
Integrazione: Verifica della collaborazione fra più oggetti nel formare un sottosistema, al fine di verificare il comportamento esterno e i contratti di interfaccia

Verifica e validazione: Controllo del soddisfacimento dei requisiti funzionali

- Fit (http://fit.c2.com/)
- Fitnesse (http://fitnesse.org/)

Ambiente di lavoro

> Tipi di automazioni



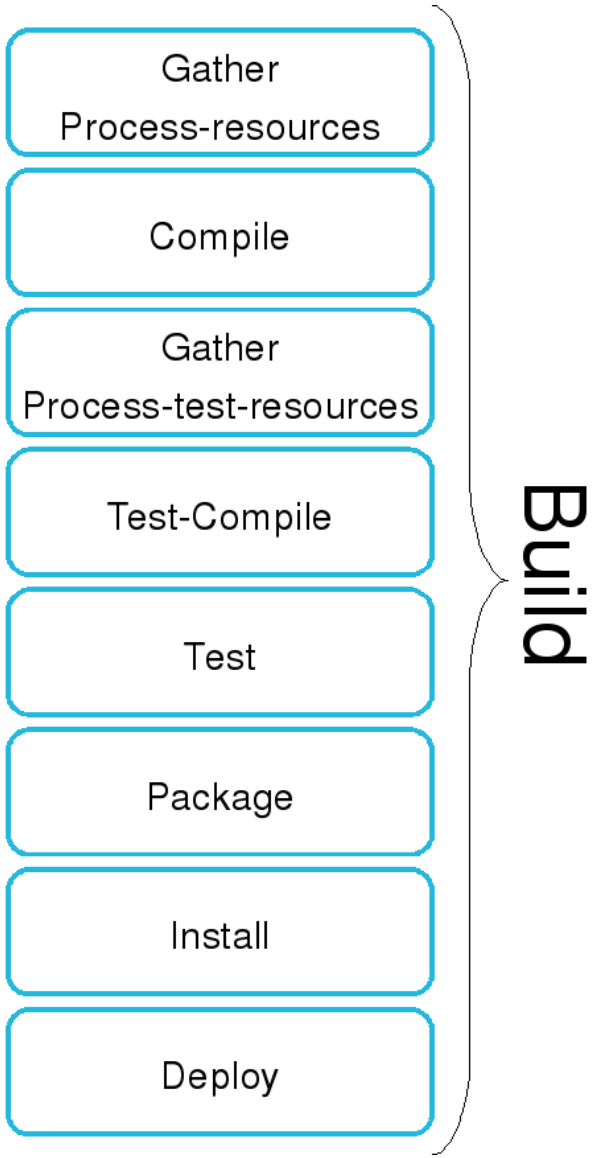
- **Comanded Automation:** Eseguire un comando che effettua un insieme di attività (p. es. .sh, .bat)
- **Sheduled Automation:** *Commanded Automation* che vengono eseguite ad intervalli di tempo prestabiliti (p. es. cron, at)
- **Triggered Automation:** *Commanded Automation* che vengono eseguite quando accade un evento (p. es. *commit* del codice sorgente → processo di *build*)

Ambiente di lavoro

> La costruzione del *software* (*build*)

Caratteristiche del processo

- **Completo:** Indipendente da fonti non specificate nello *script* di *build*
- **Ripetibile:** Accede ai file contenuti nel sistema di gestione del codice sorgente: una esecuzione ripetuta dà lo stesso risultato
- **Informativo:** Fornisce informazioni sullo stato del prodotto
- **Programmabile:** Può essere programmato ad una certa ora e fatto eseguire automaticamente
- **Portabile:** Indipendente il più possibile dall'ambiente di esecuzione



Ambiente di lavoro

> Strumenti di *project automation*

Automazione di tutte le attività definite e ripetibili

- Riduzione errori
- Maggiore documentazione

- Ant (http://ant.apache.org/)
  - Linguaggio di *script*
  - Semplicità d'uso
- Maven (http://maven.apache.org/)
  - Gestore di progetti (processo di *build*, documentazione, gestione dipendenze ...)
  - Si basa su delle convenzioni
  - Rende standard la gestione di progetti

Ambiente di lavoro

> Strumenti di analisi del codice

- Imporre il rispetto di convenzioni e stili
- Verificare la congruità della documentazione
- Controllare metriche ed indicatori (complessità ciclomatica, grafo delle dipendenze, numerosità delle linee di codice)
- Ricercare codice copiato in più punti
- Ricercare errori comuni nel codice
- Misurare la percentuale di codice testato
- Ricercare indicatori di parti incomplete (p. es. tag)

http://checkstyle.sourceforge.net/

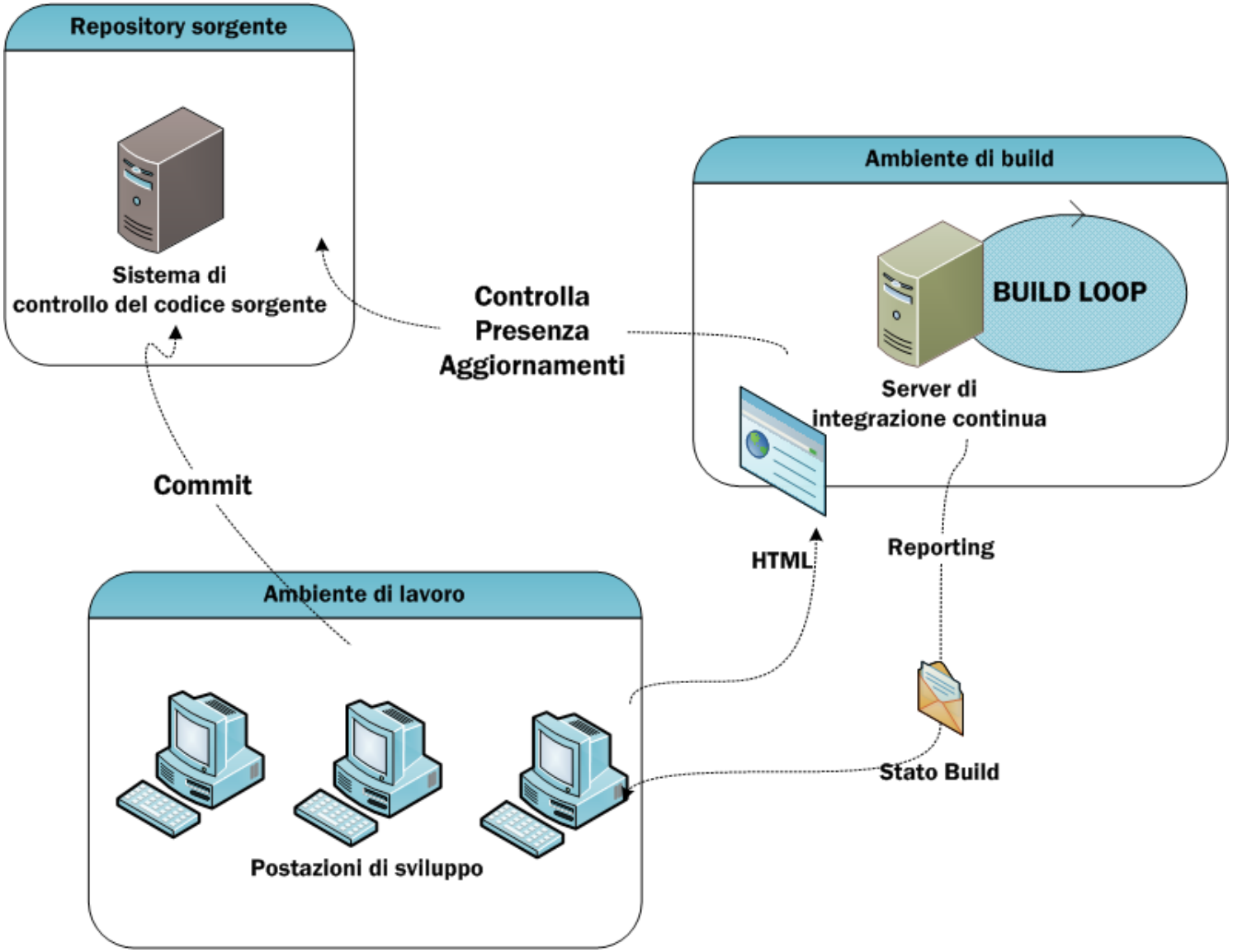
http://pmd.sourceforge.net/

http://findbugs.sourceforge.net/

http://cobertura.sourceforge.net/

Visione in dettaglio

> Ambiente di *build*



Ambiente di *build*

> Integrazione continua 1/2

- Ogni volta che una attività è completata viene creato il prodotto ed eseguiti tutti i test
- Lo scopo dell'integrazione continua è assicurare che il prodotto sia in ogni momento in uno stato consistente
- Se il processo di costruzione fallisce l'attività non continua fino a che il prodotto non viene riparato
- Richiede che il processo di costruzione sia relativamente veloce

Ambiente di *build*

> Integrazione continua 2/2

Benefici:

- Stabilità del prodotto e del processo di costruzione e verifica
- Aumento dell'attività di verifica
- Aumento delle informazioni sullo stato del prodotto
- Facile individuazione e isolamento degli errori
- Semplificazione dell'attività di messa in opera del prodotto

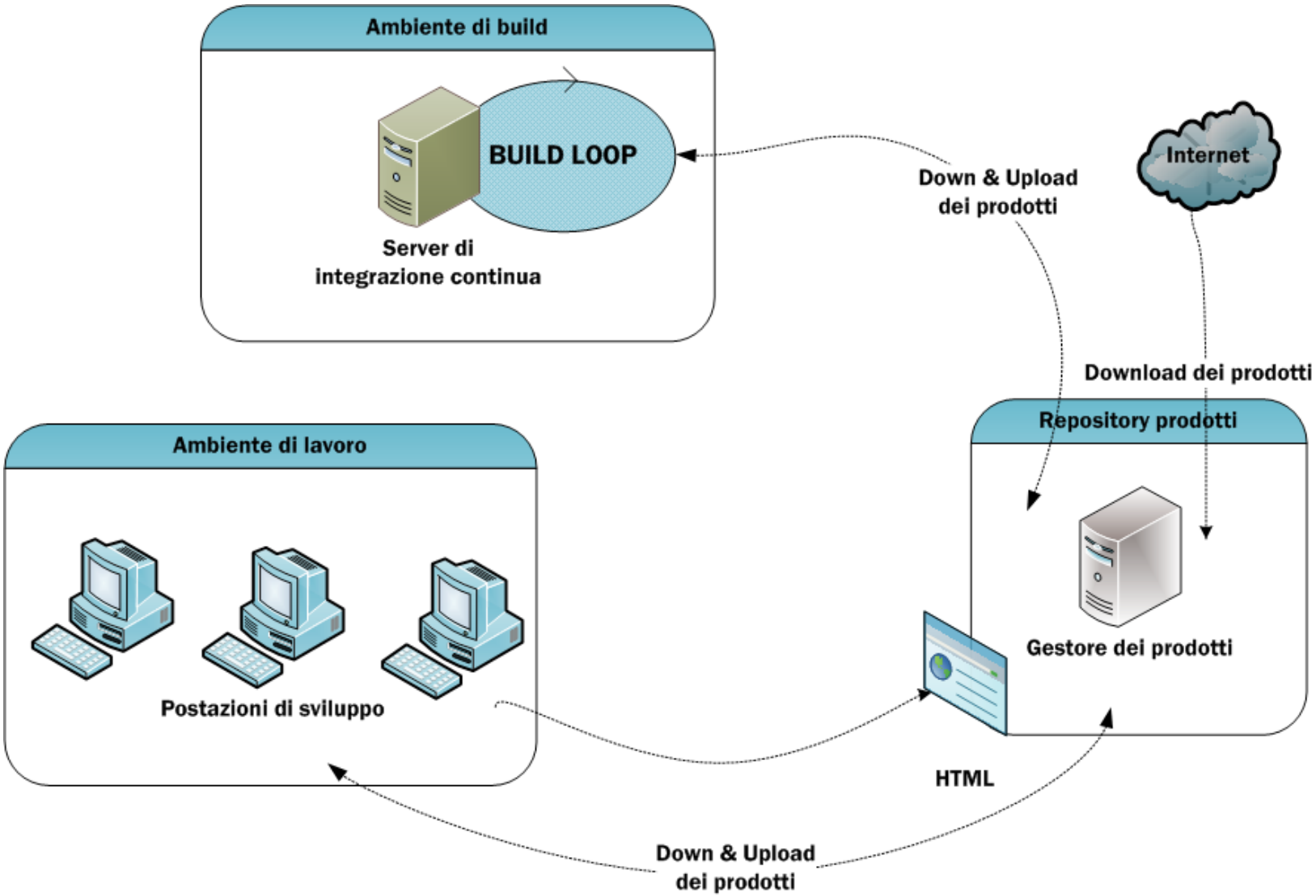
Ambiente di *build*

> Strumenti

- Cruise Control (http://cruisecontrol.sourceforge.net/)
  - Più adatto per essere utilizzato con ANT
  - Configurazione attraverso file *XML*
- Continuum(http://continuum.apache.org/)
  - Più adatto per essere utilizzato con MAVEN
  - Configurazione tramite *GUI*

Visione in dettaglio

> *Repository* Prodotti



Repository prodotti

> Gestore dei prodotti

- Unico luogo dove depositare e pubblicare i prodotti
- Permette di gestire e associare permessi d'accesso sui prodotti
- Agisce da intermediario per scaricare prodotti da depositi esterni
- Permette di effettuare ricerche, navigare e reperire informazioni riguardanti i prodotti
- Permette di effettuare operazioni di ordinaria manutenzione sui permessi e sui prodotti

Repository prodotti

> Strumenti

- Archiva (http://archiva.apache.org/)
- Artifactory (http://www.jfrog.org/sites/artifactory/1.2/)

> Sommario

1. Contesto e obbiettivi
2. Processo
3. Conclusioni
4. Esempi

Conclusioni

> Processo per migliorare la qualità di prodotto

- ✓ Risultati garantiti nel lungo periodo
- ✓ Benefici incrementali
  - Si ottengono vantaggi anche da una applicazione parziale o progressiva
- ✘ Tempo di avvio lungo
- ✘ Gestione complessa di molte componenti

Manuale che descrive gli strumenti che permettono di realizzare il processo

> Sommario

1. Contesto e obbiettivi
2. Processo
3. Conclusioni
4. Esempi