JCIS3:一個支援跨平台專案的持續整合系統
JCIS3: A Continuous Integration System for Cross-Platform Projects

張勝雄　陳建村*　鄭有進　徐天送　謝金雲
國立臺北科技大學資訊工程系，美超微電腦股份有限公司*
Department of Computer Science and Information Engineering
National Taipei University of Technology
Super Micro Computer, Inc*
Sheng-Hsiung Chang, Chien-Tsun Chen*, Yu Chin Cheng, Tien-Song Hsu, Chin-Yun Hsieh
Email: { t6598035, yccheng, s459902, hsieh}@ntut.edu.tw, teddyc@supermicro.com.tw*

摘　要

隨著電腦硬體與程式語言的快速演進，現今的軟體系統需要支援多個不同軟硬體平台已經逐漸成為常態。本論文之目的，在於探討持續整合系統要如何支援跨平台軟體專案的開發。由於現有的持續整合系統，大多沒有考慮到跨平台專案的因素，因此必須針對每一個平台個別安裝一套系統，造成使用與管理上的困難。在本論文中，我們將介紹一個支援跨平台專案的持續整合系統－JCIS3。我們首先區分*整合工作*與*整合環境*(也就是軟體專案所支援的平台)，並針對每一個整合工作描述其所需要的整合環境，達到自動分派跨平台整合工作之目的。所有跨平台整合結果將統一呈現在一個報表畫面中，而當軟體專案所需支援的平台增多時，這些平台可以動態的加入到 JCIS3 之中，大幅簡化管理與維護的工作。

關鍵詞：跨平台專案、持續整合

一、前言

隨著電腦軟體與硬體快速發展，在現今一個軟體系統需要同時支援多個不同的軟硬體平台已成為常態。例如 Apache Tomcat 網頁伺服器、FireFox 網頁瀏覽器以及 Oracle 資料庫伺服器都同時支援 Windows 與 Linux 作業系統。對於企業或軟體開發者而言，提供跨平台的軟體，就表示擁有更多潛在的客戶群與市場機會。然而，開發跨平台軟體並不是一件容易的工作，即使軟體開發團隊選擇了跨平台的程式語言與函式庫(例如 Java、ANSI C、Python、Qt)，所開發出來的軟體也需要逐一地在每一個平台上被整合與測試，方可確保軟體能在不同平台上正常執行。若這些跨平台的整合與測試工作無法被自動化執行，則開發人員必須為每種平台手動執行測試，這將是十分耗費時間與人力且容易發生錯誤的工作。如果軟體在開發期間能在其所支援的所有平台環境中經常地自動執行整合與測試，則可早期發現與平台相依性有關的錯誤並加以修正，提升軟體系統的品質並降低軟體生命週期的開發成本。

本論文探討應用持續整合(continuous integration)系統於上述跨平台軟體專案可能遭遇的問題，並提出解決方案。持續整合的目的，便是以自動化的方式，定期（且頻繁地）執行整合與測試工作，以達到「早期發現（軟體缺陷），早期治療」的效果。目前市面上已有許多持續整合軟體，較知名的有 ThoughWorks 公司提供的 CruiseControl[1]、Apache 組織所開發的 Gump[2]與 Continuum[3]、商業公司 Urbancode 的 Anthill OS[4]。基本上這些系統本身並沒有同時支援多個平台的機制，若要使用這些系統來建置跨平台的持續整合環境，則必須針對每一個平台，安裝一套持續整合系統。例如，如果要支援 Windows XP 與 Fedora，

則必須在這兩個平台上分別安裝一套持續整合系統。當所支援的平台越多時，就必須建置很多套系統，管理與維護都相當不易。在本研

如圖 1 所示，在分散式持續整合流程中主要分成三種參與角色：程式開發人員(developer)、測試人員(Tester)、以及專案經理

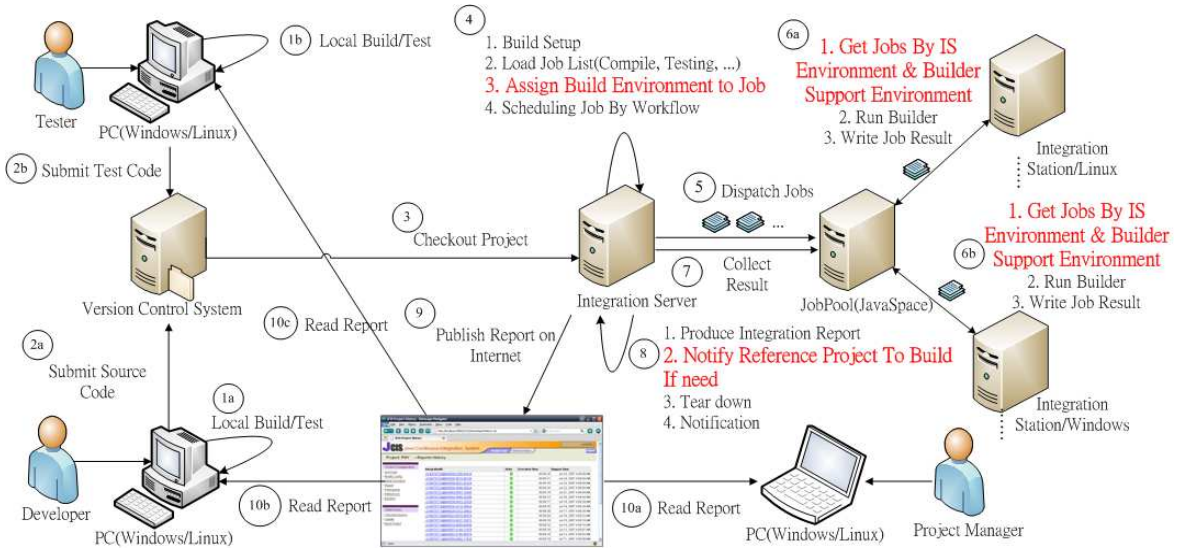

圖 1 持續整合運作流程圖

究中，我們擴充 JCIS2 [5]（一個開放原始碼的持續整合系統），實作一個支援跨平台專案的持續整合系統－JCIS3。藉由描述不同整合工作（例如，編譯、單元測試）所需要執行的平台環境，JCIS3 能自動將這些整合工作分派到所需的平台環境執行。整合後的結果將統一顯示在一個報表畫面中，使用者透過單一的整合結果報表便可得知專案在不同平台的整合是否成功。當軟體專案所需支援的平台增多時，這些平台可以動態的加入到 JCIS3 之中，只需透過修改設定檔案便可增加對於不同平台的支援，大幅簡化持續整合系統的管理與維護工作。

二、持續整合運作概念

在先前的研究[5]中，歐伯浩等人提出一個分散式持續整合應有的環境與運作流程，並利用 JINI[6]的 JavaSpaces[7]服務實作一個分散式持續整合系統。本研究將修改部分運作流程與設計以支援跨平台專案整合與測試的能力。

(Project Manager)。他們分別在各自的電腦上進行軟體開發、測試及專案管理。在專案開發初期，專案經理會在持續整合系統上建立專案，並且設定定時建構、要執行的整合工作及整合工作執行的整合環境。隨著開發的進行，程式開發人員及測試人員會將程式碼與測試碼簽入(commit)版本控制系統(version control system)中，而持續整合系統會依照專案經理先前的設定進行整合。當整合執行完畢後，系統會通知相關人員建合的結果。以下細部說明持續整合的運作流程。

1a. 程式開發人員在本地端電腦上開發程式，並進行編譯及測試 (local build & testing)。

2a. 程式開發人員將通過單元測試的程式碼簽入到一個專案共用的版本控制系統。

1b. 測試人員從版本控制系統中取出專案最新版本的原始碼，接著撰寫測試案例(Test Case)，最後測試程式。

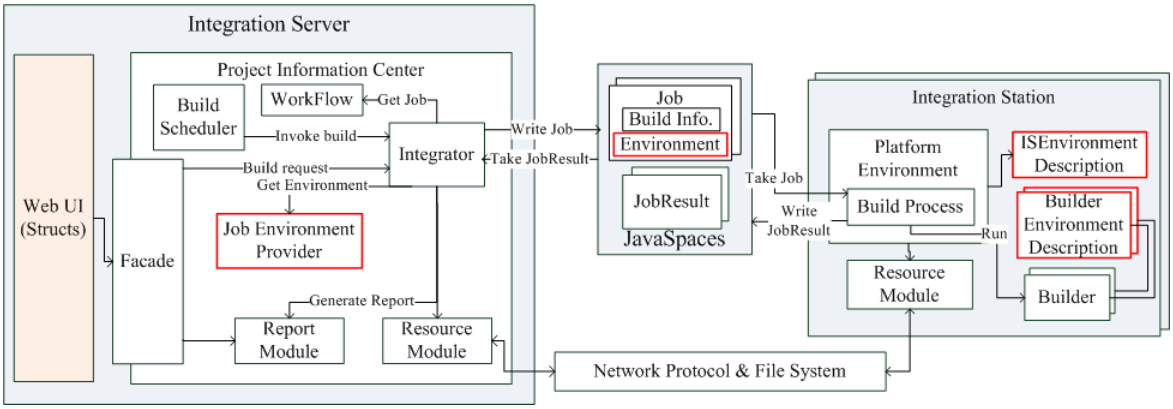

圖 2 系統架構圖

2b. 測試人員確認所有撰寫的測試案例通過後，將測試碼簽入版本控制系統。

3. Integration Server 定時地到版本控制系統取出目前最新版本的原始碼和測試碼，並依照相關設定進行專案建構(build)。

4. 一系列整合活動以一個可擴充的整合工作流程(build workflow)控制工作的執行順序，可依據專案需求不同而設定。在JCIS中包含對於原始碼分析、產生文件檔、包裝可執行檔等建構支援，其中最基本的整合工作包含原始碼的編譯與測試。

5~7. (5)Integration Server依據專案設定的整合流程及整合工作所指定的整合環境，將整合工作分派到JobPool中，直到整合流程中的所有工作都執行完畢，代表建構結束。(6a)(6b)遠端的Integration Station依據本身平台環境及Builder能支援的平台環境之條件，到JobPool中將與環境符合的工作取回執行。(7)最後將執行結果放入JobPool，讓Integrator Server取回並合併、產生所有的整合報表。

8. Integrator Server產生建構報表，並結束此建構活動。若有相依性專案[8](reference project)時，則通知其他相依的專案建構。最後通知專案相關人員建構完成。

9. 完成建構後，JCIS將產生的結果，包含目的檔、文件檔與分析報表放到Web Server上，以提供專案相關人員瀏覽。

10. (10a)專案經理、(10b)程式開發人員與(10c)測試人員透過瀏覽器觀看建構結果。

三、系統架構與設計

3.1 系統架構

圖 2 為 JCIS 的系統架構圖。系統主要分成三個部分：Integration Server (以下簡稱 Server)、JavaSpaces、以及 Integration Station(以下簡稱 IS)。Server 主要負責專案管理、控制整合建構流程、以及使用者請求處理。JavaSpaces 為 JINI 的服務之一，是一種支援以物件為基礎的處擬共享記憶體架構，在 JCIS 中扮演 Server 與 IS 間傳遞 Job 與 JobResult 的暫存容器。Server 將每一個整合工作以 Job 封裝所有整合工作的資訊並放到 JavaSpaces 中，使 IS 能夠取得整合工作。IS 為一個佈署於遠端機器上的 JCIS Client，負責將 Server 分派的整合工作從 JavaSpaces 中取回，接著在本地端執行整合工作。工作執行結束後，以 JobResult 封裝其整合結果寫入 JavaSpaces，且一併將相關的 Artifacts 壓縮上傳給 Server。

當我們建構一個跨平台專案時，Façade 介面收到建構訊息後，呼叫 Integrator 負責整個建構過程。Integrator 向 WorkFlow 取得整合工作後，透過 Job Environment Provider 設定每一個整合工作要執行的整合環境，最後放到 JavaSpaces 中。在 IS 部分，Platform Environment 讀取 IS Environment Description(平台環境描述)與 Builder Environment Description(Builder 的環境描述)後，由 Build Process 取得與平台環境符合的整合工作。Build Process 執行相對應的 Builder 並產生整合報表，再回報給 Integration Server。當所有的整合工作都執行完成，代表專案建構完成。

3.2 跨平台設計

基於歐伯浩等人的研究[5]，我們擴充其系統設計，以支援跨平台專案的建構活動。由於跨平台專案的整合工作需要分派到指定的平台環境上執行，因此我們為每種整合工作定義執行的整合環境描述和 IS 能取得與佈署的平台環境相符的整合工作來解決工作分派到指定平台環境的問題，並分成兩小節詳細說明其設計概念。

3.2.1 整合工作執行的整合環境描述

當專案建構時，Integrator 將這些整合工作依照整合流程(workflow)依序放到 JavaSpaces 裡，讓遠端 IS 取得整合工作。由於整合工作沒有指定其執行的整合環境，因此任何遠端的 IS 皆能夠取回執行。對於跨平台專案建構而言，該做法無法確保每一個整合工作會在何種整合環境上執行。因此若要支援跨平台專案建構，我們提出的作法是為每一個整合工作定義其整合環境描述，指定整合工作執行的整合環境。

我們首先定義整合工作執行的整合環境描述，主要的區別在於相同的整合工作在不同整合環境上執行對專案建構的實質意義。有些整合工作像編譯、測試等，會考量它是在何種整合環境上執行。例如跨平台專案在 Windows XP 與 Fedora 兩種環境上執行測試，因為專案測試的平台環境不同，所以視為相異的整合工作，我們稱這種類型的工作**與整合環境有關**。專案建構時，JCIS 使用者必須為這類的工作設定其執行的整合環境；另一種為**與整合環境無關**的整合工作，例如統計專案的各種行數分析(總行數、註解行數等)、code coverage 分析等，無論是在何種整合環境上執行，則建構結果都

表 1 JCIS 系統中的整合工作能支援的整合環境

整合工作的種類	Builders	支援的整合環境	
		Operating System	Library
Compile	Java Compile Builder	Windows / Linux 註 1	JDK 1.5/.1.6+
	NAnt Compile Builder	Windows	VisualStudio 2003/2008
	GCC Compile Builder	Windows / Linux	GCC 3.2/3.3+
Testing	AntJUnit Builder	Windows / Linux	JUnit 3.0/4.0+
	CppUnit Builder	Windows / Linux	CppUnit 1.11.0+
Code Analysis or Other	StatSVN Builder	Windows / Linux	無
	Code Coverage Builder	Windows / Linux	無
	LOC Builder	Windows / Linux	無
註 1：Windows / Linux 代表兩種平台系列的作業系統皆可			

是一樣的，且對專案的意義亦是相同的。

表 1 為 JCIS 系統中的整合工作能支援的整合環境。依整合工作的性質，我們主要可區分成 Compile、Testing、Code Analysis 三種類型。由於 Code Analysis 或其他的整合工作，因整合環境的不同並不會改變其整合結果，因此我們不需為此類型的整合工作考慮其整合環境。Compile 與 Testing 這兩類的整合工作因專案開發語言不同，有相對應的 Builder 與能執行的整合環境，例如 Java Compile Builder 能編譯用 Java 語言開發的專案，其執行的整合環境可為 Windows / Linux 與 JDK 1.5+之組合。而 Testing 這類的 Builder 之整合環境可為 Windows/Linux 與單元測試所使用的框架(framework)版本之組合。

3.2.2 Integration Station(IS)能取得與佈署的平台環境相符的整合工作

每一個 IS 能執行哪些整合工作主要依據 Builder 能在哪些平台環境上執行。有些 Builder 執行需要在平台環境上額外安裝函式庫或軟體，例如 StatSVN Builder 需安裝 Subversion；有些則需要 Third-party 的可執行程式，例如 LOC Builder 使用 AStyle[9]外部程式。再者，有些 Builder 僅能在特定平台環境上執行，如表 1 NAnt Compile Builder 只能在 Windows 系列的作業系統上執行。這些平台環境與 Builder 支援環境的相關資訊都與 IS 能執行哪些整合工作息息相關。因此我們分成平台環境描述與 Builder 的環境描述兩部份說明如何使 IS 取得與平台環境相符的整合工作。

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tns:is_environment_config xmlns:tns="http://www.example.o
3    <tns:ISEnvironment>
4      <tns:operatingsystem>                                    1
5        <tns:property key="bit" value="32" />
6        <tns:property key="name" value="WindowsXP" />
7        <tns:property key="version" value="professional" />
8      </tns:operatingsystem>
9      <tns:installsoftware>                                    2
10       <tns:software_item name="jdk" version="1.6">
11         <tns:property key="descirption" value="java develo
12       </tns:software_item>
13       <tns:software_item name="subverion" version="1.4.6">
14         <tns:property key="descirption" value="java develo
15       </tns:software_item>
16       <tns:software_item name="Visual Studio2008" version=
17         <tns:property key="descirption" value="java develo
18       </tns:software_item>
19     </tns:installsoftware>
20   </tns:ISEnvironment>
21 </tns:is_environment_config>
```

圖 3 IS 佈署的平台環境描述

- 平台環境描述

平台環境是指實體機器上所安裝的作業系統、函式庫及軟體的總稱。其環境描述內容分成兩個部分：作業系統與平台上所安裝的函式庫或軟體。作業系統的描述包含幾位元、作業系統名稱、版本三個屬性；而平台安裝的函式庫或軟體之描述包含名稱和版本兩種屬性。圖 3 為 IS 佈署的平台環境描述檔，(1)說明 IS 佈署於 32 位元的 Windows XP Professional 的平台上(4~7 行)。(2)平台環境上已安裝 JDK 1.6(10~12 行)、Subversion 1.4.6(13~15 行)、以及 Visual Studio 2008(16~18 行)，IS 透過平台環境描述可知道目前佈署於何種環境上，可作為挑選整合工作的條件。

- Builder 的環境描述

Builder 是整合工作的執行者，每一種整合工作都有相對應的 Builder 負責執行工作。在支援跨平台專案的情況下，由於 Builder 必須在不同平台環境上執行，因此我們定義每一種 Builder 的**必要平台環境描述**及**能支援的平台環境描述**。

Builder 必要平台環境描述方面，依 Builder 的執行方式我們區分成兩種類型：**一種是透過命令列執行指令，通常必須事先在平台環境上安裝特定的軟體或函式庫**，例如 JCIS 系統中 StatSVN Builder，是一個能夠分析專案在 Subversion 的活動情況(簽入、簽出、以及程式碼行數的變動情形)，使用前必須事先在平台上安裝 Subversion，接著透過 svn 指令取得 log

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <tns:builder_config xmlns:tns="http://www.example.org/Javac" xml
3      <tns:builder type="N/A"
4          description="Generates various tables and charts describ
5          display-name="StatSVN" classpath="ntut.csie.jcis.builder
6          name="statsvn">
7      1 <tns:requires-environment>
8            <tns:platform>                                    2
9                <tns:os_item name="N/A">
10                   <tns:property name="bit" value="N/A" />
11                   <tns:property name="version" value="N/A" />
12               </tns:os_item>
13           </tns:platform>
14           <tns:other>                                       3
15               <tns:property name="jdk" value="1.5" />
16               <tns:property name="Subversion" value="1.4.5" />
17           </tns:other>
18       </tns:requires-environment>
19   </tns:builder>
20 </tns:builder_config>
```

圖 4 StatSVN Builder 必要的平台環境描述

檔。**另一種是執行一支可執行檔或透過 JVM 執行 jar 檔**，與前者的差異是只需將 jar 檔或可執行檔加入 IS 指定的資料夾即可。此部分的資訊是告訴 Builder 的使用者，在執行 Builder 之前，需要做哪些環境上的設定；而 Builder 能支援的平台環境描述部分，描述此 Builder 能執行(支援)的平台環境。由於 JCIS 可外掛自行開發的 Builder，因此每一個 Builder 的加入都必須包含 Builder 的 jar 檔及設定檔，設定檔的內容包含 Builder 的環境描述與偏好設定。環境描述的內容包含：

- 必要的平台環境描述：此環境描述代表執行此 Builder 時，平台環境上最少需要安裝這些作業系統、函式庫或軟體等。圖 4 為 StatSVN Builder 必要的平台環境描述，(1)StatSVN Builder 必要的平台環境描述區塊(7~18 行)。(2)Platform 的屬性皆為 N/A，因此沒有限定需何種作業系統(8~13 行)。(3)平台環境上需安裝 JDK 1.5 及 Subversion 1.4.5(14~17 行)。
- 能支援的平台環境描述：此環境描述代表 Builder 能在哪些平台環境上執行，換言之，就是限制 Builder 僅能在適合的平台環境上執行。圖 5 為 NAnt Compile Builder 支援的環境描述，NAnt 是一個能編譯 C#原始碼的 Builder，僅能支援 Windows 的作業系統之平台環境。(1) NAnt 能支援的平台環境描述區塊(7~25 行)。(2)NAnt 僅能支援 32bit Windows XP Professional 及 WindowsServer2003 的作業系統的平台環境(8~19 行)。(3)NAnt 能支援 VS2008 版本編譯原始碼(20~24 行)。

我們透過平台環境描述及 Builder 能支援的平台環境描述，取出兩者相符的環境描述，確保 IS 能取得與平台環境及 Builder 支援的環境相符的整合工作，例如 IS 佈署的平台環境上有安裝 Visual Studio 2008(請參考圖 3)，若此平台上有支援 NAnt Compile Builder(請參考圖 5)，則此 IS 能以 Visual Studio 2008 版本執行 C#語言的編譯工作。

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <builder_config xmlns="http://www.example.org/Javac">
3      <builder type="C#"
4          description="Build dot net platform project"
5          display-name="NAnt" classpath="ntut.csie.jcis.bu
6          name="NAnt">
7      1 <support-environment>
8            <platform>
9                <os_item name="WindowsXP">                  2
10                   <property name="bit" value="32" />
11                   <property name="version"
12                             value="professional" />
13               </os_item>
14               <os_item name="WindowsServer">
15                   <property name="bit" value="32" />
16                   <property name="version"
17                             value="2003" />
18               </os_item>
19           </platform>
20           <other>                                         3
21               <property name="VisualStudio2008"
22                         value="Professional" />
23               </property>
24           </other>
25       </support-environment>
26   </builder>
27 </builder_config>
```

圖 5 NAnt Compile Builder 能支援的環境描述

四、應用範例

跨平台專案的應用範例

我們以開放原始碼專案 SyncFree [10]作為跨平台測試的專案。SyncFree 是一個開放原始碼的個人資料同步軟體，其開發目的為解決日益複雜的個人資料同步問題。當 SyncFree 在進行檔案同步時，需要使用檔案名稱和路徑。由於不同的作業系統上，其檔案命名長度

限制與路徑格式不同，這些因素會影響檔案同步的正確性。因此，若要讓 SyncFree 能在跨平台環境上正常執行功能，則專案開發的測試流程就必須包含跨平台的測試案例，如此才能驗證在跨平台環境中 SyncFree 的檔案同步功能之正確性。

SyncFree 專案中包含原始碼及事先寫好的測試案例，並且事先佈署 SyncFree 的測試資料與測試設定檔放到各平台指定的目錄下，如圖 6 為 SyncFree 的跨平台測試示意圖。Integration Server 從 SVN Server 將 SyncFree 專案取出後，把原始碼、測試碼、及相關函式庫資源依據事先設定的建構流程分派到四種平台執行編譯與測試，最後得到總測試報圖。

圖 7 為 SyncFree 專案的 Builder 列表，每一個 Builder 代表一種整合工作，並指到其整合環境。列表中每個平台包含 Java Compile、JUnit Testing、兩種整合工作。

圖 8 為 SyncFree 的跨平台的總整合報表。每一列代表一個整合工作，其中包含整合狀態、整合環境、建構時間等相關資訊。最後兩個欄位為較詳細的建構記錄與 Artifacts 連結。在這個統一的報表頁面中，我們可觀察到 SyncFree 在四個平台的編譯工作與測試工作都是成功的，便可立即得知 SyncFree 的檔案同步功能在這四個平台上能正常運作。

五、相關研究

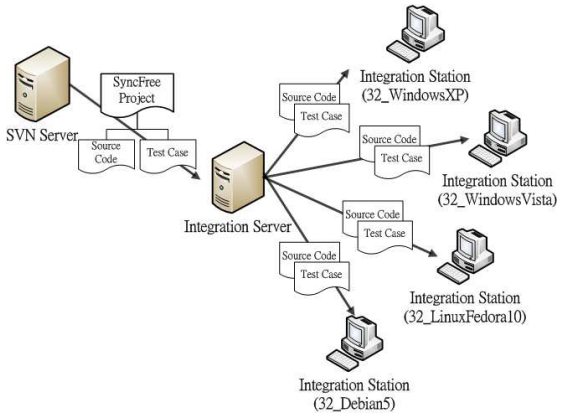

圖 6 SyncFree 跨平台測試示意圖

Builder List — Add builder

Builder Name	Build OS	Description	Action
JUnit Testing	Linux_Fedora10_32		
Javac	Linux_Debian5_32		
JUnit Testing	Linux_Debian5_32		
Javac	Linux_Fedora10_32		
Javac	WindowsVista_Ultimate_32		
Javac	WindowsXP_professional_32		
JUnit Testing	WindowsVista_Ultimate_32		
JUnit Testing	WindowsXP_professional_32		

圖 7 SyncFree 專案的 Builder 列表

先前吳家豪[11]等人研究中，以 IP Address 指定整合工作到不同的整合環境上執行。只要

Project Build Summary

- Build Start Date : 2009/04/08 08:09:03
- Build End Date : 2009/04/08 08:26:29
- Build State : Success
- Source Version : 1253

ProjectName	Builder	State	BuildEnvironment	IntegrationStation info.	Duration	Report	Build Result
SyncFree	compile	✓	Linux_Debian5_32	IntegrationStationIP: debian/127.0.1.1	0h 1m 30s 753ms	View Report	
SyncFree	compile	✓	Linux_Fedora10_32	IntegrationStationIP: localhost.localdomain/127.0.0.1	0h 1m 46s 879ms	View Report	
SyncFree	antJUnit	✓	Linux_Debian5_32	IntegrationStationIP: debian/127.0.1.1	0h 0m 1s 335ms	View Report	View Result
SyncFree	antJUnit	✓	Linux_Fedora10_32	IntegrationStationIP: localhost.localdomain/127.0.0.1	0h 0m 8s 736ms	View Report	View Result
SyncFree	compile	✓	WindowsVista_Ultimate_32	IntegrationStationIP: lab1321-PC/192.168.238.131	0h 2m 45s 906ms	View Report	
SyncFree	antJUnit	✓	WindowsVista_Ultimate_32	IntegrationStationIP: lab1321-PC/192.168.238.131	0h 0m 10s 500ms	View Report	View Result
SyncFree	compile	✓	WindowsXP_professional_32	IntegrationStationIP: ntut-c3b010d4b4/192.168.44.135	0h 3m 44s 468ms	View Report	
SyncFree	antJUnit	✓	WindowsXP_professional_32	IntegrationStationIP: ntut-c3b010d4b4/192.168.44.135	0h 0m 14s 890ms	View Report	View Result

圖 8 SyncFree 跨平台測試報表

建立一套系統，便可達到跨平台整合的目標。但是每一個整合工作所包含的資訊中，沒有整合環境的描述，必須由 JCIS 使用者自行設定每組 IP Address 與整合環境之對應，當整合環境愈來愈多時，使用上有不便之處。

在歐伯浩[5]等人的研究中，使用 JavaSpaces 服務改善了整合工作以 IP Address 指定整合環境的方式，讓每台建構機器自由競爭整合工作，並且提升系統的整合效能。然而，整合工作仍然無整合環境相關的描述，也就產生許多問題，例如：無法確保整合工作會在預期的整合環境上執行、有些整合工作相依於整合環境才能執行。因此若要支援跨平台整合建構，必須對整合工作定義執行的整合環境描述。

目前市面上有許多持續整合系統陸續被開放完成，我們檢視幾個開放原始碼持續整合系統，Anthill、Continuum、CruiseControl、Gump。表 2 為現存持續整合系統與 JCIS3 之功能比較。

- Anthill OS：由商業公司 Urbancode 所提供的開放原始碼持續整合系統，以建構工具 Ant 為基礎。此系統安裝設定都非常簡易，但沒有一個統整性的報告瀏覽介面，不支援將專案分派至不同電腦整合的功能。
- Continuum：Apache 的整合工具。此工具可支援 Maven1、Maven2 與 Ant，安裝與配置較簡易。並且有 Web 介面可提供專案內容的修改。不支援跨平台專案分派到不同平台環境整合，必須為每一個平台安裝一套持續整合系統。
- CruiseControl：CruiseControl 是由 ThoughtWork 所提供的開放原始碼持續整合系統。CruiseControl 可支援 Ant[12]、NAnt[13]、Maven[14]…等建

表 2 現存持續整合系統與 JCIS3 之功能比較

功能/特色	CruiseControl	Anthill OS	Gump	Continuum	JCIS2	JCIS3
開放原始碼	是	否	是	是	是	是
免費使用	✓	✗	✓	✓	✓	✓
開發語言	Java	Java	Java	Java	Java	Java
Build Management						
支援跨平台專案建構	✓註 1	✓註 1	✗	✗	✗	✓註 2
程式庫相依性	✓	✓	✓	✓	✓	✓
同時建構多個專案	✓	✓	✗	✓	✓	✓
分散式架構並行執行	✓註 1	✓註 1	✗	✗	✓	✓
Build Report						
跨平台建構報表	✗	✗	✗	✗	✗	✓
Support						
使用建構工作	Ant, Maven, NAnt	Ant	Ant	Ant, Maven	Ant 與自行定義	Ant 與自行定義

註 1：每個平台上都安裝一套持續整合系統。當每個平台上的專案建構完成後，將報表及相關 logs publish 至同一個 Web server。

註 2：一台電腦上安裝 JCIS Server，其他電腦安裝不同平台並且佈署 JCIS Client 程式。專案建構時，由 JCIS Server 分派建構工作到不同平台的電腦上執行。

構，並提供基本的建構流程與測試流程。在支援跨平台編譯、測試方面，使用者必須在多台電腦上各自安裝並設定一套 CruiseControl 系統[15]。專案在本地端系統執行編譯、測試後，系統將 Build Logs 與 Artifact 透過檔案共享的功能，發佈到同一個 CruiseControl 上，讓使用者可透過此 CruiseControl 的 DashBoard 觀看其他不同平台上 CruiseControl 的專案建置結果。此種做法使得使用者必須維護多個持續整合系統，使用並不方便。

- Gump：Apache 提出的另一個持續整合工具，可支援 Ant、Maven 等多種建構工具。可提供專案間相依性維護的支援。不支援跨平台專案分派到不同平台環境整合，必須為每一個平台安裝一套持續整合系統。

六、結論與未來展望

本研究基於[5]過去所開發的持續整合系統－JCIS2，擴充、修改系統的設計以支援跨平台專案建構－JCIS3，能支援 C/C++、Java 語言的專案跨平台測試。

未來發展方向包含將相依性的建構工作分派到同一個平台上執行，以減少專案在建構時檔案資源上傳與下載次數、加入虛擬化技術能夠自動佈署各種不同的整合環境、以及能夠區別將不同平台的測試案例能放在相對應的環境上執行測試。

致謝

本研究由國科會計劃 NSC97-2218-E-027-015 補助，特此致謝。

參考文獻

[1] CruiseControl, http://cruisecontrol.sourceforge.net/.

[2] Apache Gump, http://gump.apache.org/.

[3] Apache Continuum, http://maven.apache.org/continuum/.

[4] Anthill OS, http://www.anthillpro.com/html/products/anthillos/default.html/.

[5] 歐伯浩、陳建村、鄭有進、徐天送, “JCIS2: 一個分散式持續整合系統 “, 第四屆台灣軟體工程研討會論文集。台南，2008年6月13~14日.

[6] JINI, http://java.sun.com/developer/products/jini/index.jsp/.

[7] JavaSpaces, http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/.

[8] 吳家豪, JCIS: 支援JAVA應用程式發展的持續整合系統－JCIS, 95/06.

[9] AStyle, http://astyle.sourceforge.net/astyle.html/.

[10] 謝金雲, SyncFree: 一個使用Java技術開發之開放原始碼個人資料同步, 國科會自由軟體專案研究計劃, 計畫編號 92-2218-E-027-020.

[11] 吳家豪、佘翠瑱、陳建村、鄭有進, “JCIS: 支援平台相依性建構之Java 持續整合系統 ”, 第三屆台灣軟體工程研討會論文集。台中，2007年6月8~9日.

[12] Apache, Ant, http://ant.apache.org/.

[13] NAnt, http://nant.sourceforge.net/.

[14] Apache Maven, http://maven.apache.org/.

CI on multiple platforms with CruiseControl, http://confluence.public.thoughtworks.org/display/CC/MultiplePlatforms/.