Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

# An Integrated Tool Architecture for Multimedia Presentation

Timothy Shih, Louis R. Chao, Chi-Ming Chung, Y.H. Wang, Y.F. Kou
Department of Information Engineering
TamKang University
Taipei 25137, Taiwan, R. O. C.

Wei-Chuan Lin
Department of Accounting and Statistics
Tak-Ming Junior College of Commerce
Taipei, Taiwan, R. O. C.
E-mail: g1320039@cs.tku.edu.tw

## Abstract

*The demonstration of multimedia presentation can be promoted by using multi-vendor's tools. The more tools are used, the more complicated communication is needed among these tools. The integration of these multimedia presentation tools is thus important. This paper describes an architecture named* ***Tool Integration Platform(TIP)*** *to integrate tools in a knowledge abstraction way.* ***TIP*** *is composed of a* ***CID(Control Integration Daemon)***, *a* ***CII(Control Integration Interface)*** *and some* ***Integration Inference Rules(IIR)*** *that are applied by the* ***Integration Inference Engine(IIE)***. *The* ***IIR*** *are stored in a* ***Repository*** *and used to deduce tool knowledge dynamically. In this way, many tools can be integrated into a cooperative multimedia presentation developing environment. To verify this architecture, a number of tools including a* ***Resource Editor***, a ***Resource Browser***, and a ***Presentation Designer*** *are integrated into* ***TIP***. *Finally, a number of* ***compared components*** are *used to assess* ***TIP*** *with other environments and standards.*

***Keywords: Tool Integration Platform, Integration Inference Engine, Integration Inference Rule, Repository, Control Integration Daemon, Control Integration Interface***

## 1. Introduction

Multimedia presentation is critical to demonstrate the effect of multimedia. The objective of each multimedia tool is to increase the productivity, provide the better view and simplify the multimedia development. Because users have their own preferred tools developed by different tool vendors, it is important to integrate those heterogeneous tools in a cooperative developing environment. To support multimedia across open distributed systems, all of the tools should have the appropriate Client/Server architecture. However, a company does not have to develop all the tools to meet users' requirements. The tools should cooperate to compensate each other's weak-points. For example, the Resource Editor(**RE**), Resource Browser(**RB**) and Presentation Designer(**PD**) tools are widely used to capture and display the multimedia resources respectively. When developing a multimedia presentation, the planner may use these tools to prepare the presentation resources. Therefore, the **RE**, **RB**, and **PD** tools should be integrated together. Thus, when the resources captured by **RE**, they are sent as the input data of the **RB** and **PD** tools automatically, and demonstrate the multimedia presentation. In this way, the job for developing multimedia presentation would be convenient. The automatic processes can be done in the same way in different multimedia developing steps via many integrated tools.

In this paper, section two describes the approaches for tool integration. Section three explains the major components of the proposed tool integration architecture called ***TIP***. Section four is partitioned into three parts. The first part describes the verification of ***TIP*** through integrating a set of multimedia tools. The second part introduces the functions and relation between the ***IIE*** and ***IIR***. The third part proposes a number of compared items for assessing the status of tools integrated in ***TIP***. Section five is the conclusion and the continuing research.

## 2. The Tool Integration Approaches

There are three evolving approaches for tool integration. The first is called ***brute-force*** approach which integrates a set of predefined tools and forms a cooperated tool environment. However, the way for exchanging data among tools is used the Import/Export functions without taking the data semantics into consideration. Therefore, there are two drawbacks in this approach. The first is that it is difficult for other tools that is not included in the pre-defined tool set to join into the integrated tools. The second is that there is no integration standard for tools to follow[5]. Thus, if there is a new tool

**tool4** which is planned to join to the integrated environment, what will be the relation between the new tool and the pre-integrated tools as drawn in Fig. 1?

The second is called ***vendor dependent*** approach which integrates a set of tools that are developed by the same tool vendor and also named as Integrated CASE(*ICASE*) tools approach. The advantage of this kind of tool integration is the tools are optimally, well-tightly integrated. The tools integrated in this approach can exchange the semantic data and provide good services. However, the semantic data cannot be exchanged among different vendor's tools. Some *ICASE* vendors are attempted to integrate with other vendor by opening their metamodel of tools. This phenomenon can be shown as Fig. 2. The well-known environments for *ICASE* tool approach are TI(Taxas Instrument)'s IEF, DEC's FUSE, and IBM's AD/CYCLE[17].

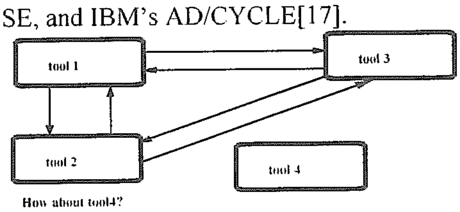


**Fig 1. The first approach**

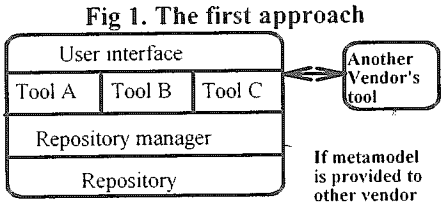


**Fig 2. The second approach**

The last is called ***vendor independent*** approach which supports the integration components such as the **Presentation**, **Process**, **Control**, and **Data Integration**. In this way, this approach provides an open extensible environment[13] and can integrate no matter what tools developed by the same or different vendor. The detailed explanations for the integration components are shown in the following paragraph.

The **Presentation Integration(PI)** is to integrate tools in a consistent Graphical User Interface(**GUI**) way. The de facto standards for **PI** are OSF/Motif, SUN/OpenWindows in UNIX, and IBM/Warp, Microsoft/Windows in DOS environment. The **Process Integration(PRI)** is to ensure an interactive tool environment to support a pre-defined process. The **Control Integration(CI)** is to provide the flexible services among tools[1,14]. The standards for **CI** are ANSI X3H6, OSF DCE, and OMG CORBA[14,21]. The **Data Integration(DI)** is to provide the data repository service for sharing the common information of tools in a consistent data format. The standards for **DI** are ECMA/PCTE's Repository, CDIF, and IEEE Std. 1175[8,9,16].

The first environment proposed to demonstrate the utility of message-server architecture is the FIELD environment developed in Brown university[19]. This environment was implemented in UNIX environment and utilized the socket library to do the Inter-Process Communication(**IPC**). The first message protocol designed to send among processes was the protocol of the FIELD environment. Products developed by HP[2] and SUN[10] are enhanced the message protocol of the FIELD environment to exchange tool messages. These products were implemented in UNIX environment and focused only in the **Control Integration**.

From the above description, each of the products or standards does not cover all the functions of integration components. As for the other standards such as PCTE, and CORBA, which include the integration components are still under discussion and revision. Therefore, this paper proposes an architecture provided not only the integration components but the ***IIE*** that applies the ***IIR*** suitably. In this way, ***IIE*** can deduce the tool knowledge dynamically and store them in the Repository.

## 3. The Platform of Tool Integration

The architecture proposes in this paper is called Tool Integration Platform(***TIP***) which can be expressed as a set of transformation functions to map a tool to other tools. After the service has been done by tools, the transformation functions can transfer control back to the original tool. The mapping is denoted in a Finite State Machine like manner as:

**TIP**=(**Q**, $\Sigma$, $\delta$, **T**, **O**), where

**Q**: A finite set of **internal states,** including **{Active Run(AR), Not Run(NR), Background Run(BR)}**

$\Sigma$: A set of input such as **{resource ...}**

**T**: A set of tools such as **{Resource Editor(RE),Resource Browser(RB),Presentation Designer(PD) ...}**

**O**: The output set of tools such as **{reviewed resource, generated presentation ...}**

$\delta$: A set of transition functions include **{provide, listen, send, notify}** and can be denoted as:

$$\delta: Q \times \Sigma \times T \rightarrow Q \times T \times O$$

For example:

$\delta_{send}$(**AR**, "resource", **RE**) $\rightarrow$ ({**BR**, **NR**},{**RB**,**PD**}, {**reviewed resource**, **generated presentation**})

This means that the running tool **RE** sends the "resource" to **RB** or **PD**. These two tools are originally in not running or background running state.

They are triggered to execute the service of reviewing the resource or generating multimedia presentation. To achieve this transformation, ***TIP*** is divided into five components which are Control Integration Daemon(***CID***), Control Integration Interface(***CII***), Integration Inference Rules(***IIR***), Integration Inference Engine(***IIE***), and the ***Repository***. With these mechanisms, ***TIP*** is a machine independent platform which can integrate any kind of multimedia tools. These mechanisms are explained as followed:

. The ***CID*** is the message server which dispatches the message to the suitable tools and triggers the ***IIE*** to apply the stored ***IIR*** suitably.

. The ***CII*** is the interface used to integrate tools into ***TIP***. There are two kinds of integration: **Tightly** and **Loosely Integration(TI and DI)**. The major difference between them is whether the source codes of tools should be modified or not. The compared result lists as followed:

| *LI* | *TI* |
|---|---|
| **CII Start** | **CII Start** |
| **Tool Invocation** | **Tool's source codes** |
| **CII Stop** | **CII Stop** |

. The ***IIE*** is the inference engine in ***TIP*** for deducing tool knowledge. It is triggered by ***CID*** when a tool is registered in ***TIP*** or a message is sent to ***CID***. In this way, the new deduced tool knowledge is produced and stored in the ***Repository***.

. The ***IIR*** are inference rules applied by ***IIE*** and stored in the ***Repository***. The ***IIR*** are the basic inference rules in ***TIP*** and can be extended by adding new ***IIR***.

. The ***Repository*** is used to store the tool knowledge and ***IIR***. The tool knowledge includes: the registered tool name, basic and extended definitions for ***IIR***, and the System Default Configuration File(SDCF) which contains the system tools.

To verify the feasibility of ***TIP***, the Microsoft's Windows system is selected as an implementation testing site. As the Microsoft's Windows system provides the **GUI** and **OLE**(Object Linking and Embedding)[12] functions to implement the **tool's presentation** and **IPC** just as the funcyions provided in UNIX environment. However, it is hard to understand and use the Application Interfaces(**API**) of **OLE**. Therefore, the ***CII*** is to assemble the **API** of **OLE** to provide an easy integration interface to encapsulate tools into ***TIP***. This architecture can also be port to the UNIX, Windows 95 or Windows NT environment.

The message protocol of ***TIP*** are followed the standard proposed by COSE[3] which are founded by HP, IBM, Informix, CDC, SUN, Digital, and SiliconGraphics companies ... etc. Therefore, the message protocol proposed in ***TIP*** may communicate with the format developed in COSE format of UNIX environment[6,11].

The implementation of ***CID*** and ***CII*** based on the Microsoft Windows system is shown in Fig. 3. The ***CID*** is an OLE-like server and the ***CII*** is to assemble the **API** of OLE.

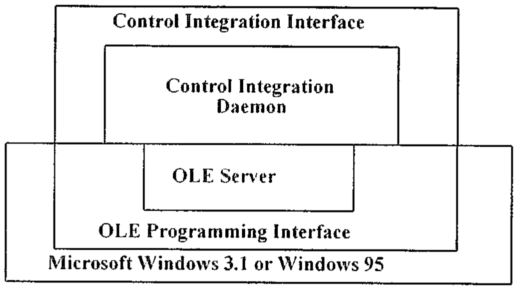


**Fig. 3 The architecture of the CII & CID**

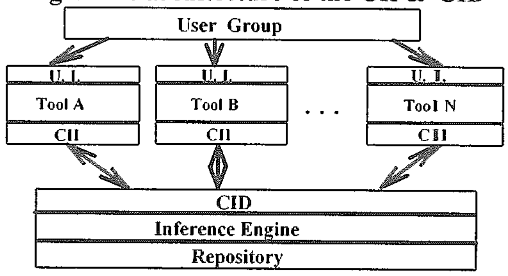


**Fig. 4 The run time scenario in *TIP***

The run time scenario is drawn in Fig. 4, in which every tool must register in the ***TIP*** through ***CII***. There are two kinds of tool registration in ***TIP***: **Static** and **Danamic Registration(SR and DR)**. As shown in the following table, the major differences between these **registration** are whether the registered tools can be removed from ***TIP*** and triggered automatically or not.

| *SR* | *DR* |
|---|---|
| **tool cannot be removed from TIP unless deleted the tool entry in SDCF** | **tool can be removed from TIP by CII** |
| **can be triggering automatically** | **can not be triggering automatically** |

The advantage of **DR** is that the tools can be easy to **plug-in and play** or **replace** into ***TIP***. That is to support the method for tools to plug-in or replace the tool which has or has not already existed in ***TIP***. In contrast, the advantage of the tool integrated in SR is that the tool can be triggered automatically by ***TIP*** when it is not in running state. However, the automatic triggering mechanism cannot take place when the tool registered in **DR**. The architecture of ***TIP*** is to enhance the standard proposed by ECMA/PCTE to provide an new integration environment[7,9] drawn as Fig. 5.

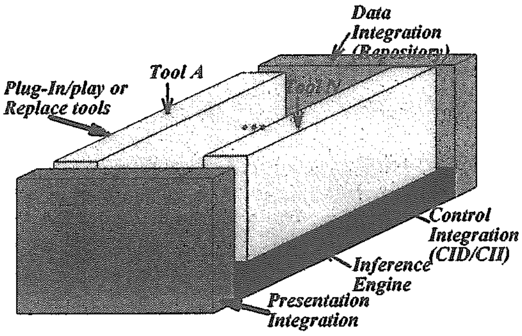


**Fig. 5 The Proposed *TIP* Architecture**

# 4. Integrated Tools, Integration Inference Rule, and Assessment for *TIP*

## 4.1 Tools integrated in *TIP*

To verify the feasibility of ***TIP***, many tools such as the Resource Editor(**RE**), Resource Browser(**RB**) and Presentation Designer(**PD**)[15,18] are integrated in this environment. Other system tools such as the tool manager(**CIP Manager**) and message monitor(**Monitor**) are also implemented in the ***TIP*** to monitor the tool invocation and the flow of message-passing. The traditional multimedia developing flow is shown in Fig. 6. The developers have to use these tools step by step to develop a multimedia presentation. Thus, these tools should be integrated for reducing developers' efforts. The **RE** includes many editors such as Text Editor, Animation Editor ... etc. to accept the digitized multimedia resources. The **RB** is used to review the resources accepted from **RE** and stored in the Resource DataBase. The **PD** is used to schedule and synchronize the presentation resources. The integration architecture in the ***TIP*** can be drawn as Fig. 7. All of these tools are triggered through the message passing to ***CID*** of ***TIP***, then, the ***CID*** drives the ***IIE*** to apply the ***IIR***[20] and deduces the tool knowledge.

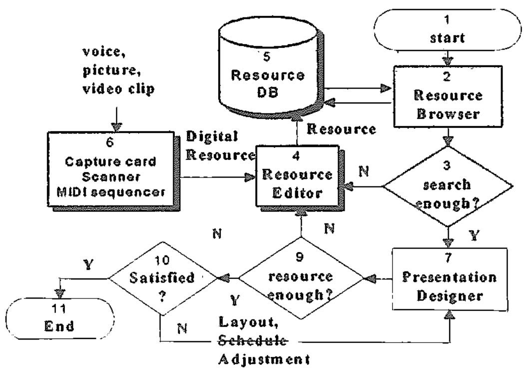


**Fig. 6 Tradition multimedia developing flow**

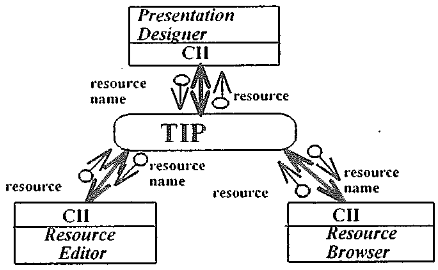


**Fig. 7 Current tools integrated in *TIP***

## 4.2 The IIE and IIR

For the sake of explaining the ***IIR*** in ***TIP***, some mathematical sets are expressed as:
Let C be the domain of input source ∋ C = {**Resource**}, T be the domain of tools ∋ T = {**RE**, **RB**, **PD**} and S be the domain of services provided by each tool in T. The integrated tools of ***TIP*** can be denoted as:

$$TIP(ToolSet)=\sum_{i,j,k} T_i(S_j(C_k))\text{ , where } \forall_i, T_i \in T,$$

$\forall_j, S_j \in S, \forall_k, C_k \in C$.

This means that the input set $C_k$ processed by the service set $S_j$. The $S_j$ is provided by the tool $T_i$. To achieve this phenomenon, some definitions, which are stored in the ***Repository***, denoting in predicate logic are:

*Definition 1*: **P(S, T)** means that tool **T** provides the service S, T∈T and S∈S. For example, the **RE** provides "Save digital resource" service. Therefore, it can be denoted as **P("Save digital resource", RE)**.

*Definition 2*: **L(S, $T_1$)** means tool $T_1$ listens the service S which is provided by other tool $T_2$, $T_1$ ,$T_2$∈T and S∈S. For example, the **RB** listens the "Save digital resource" service provided by Compiler. Therefore, it can be denoted as **L("Save digital resource", RB)**.

*Definition 3*: **I(Text, T)** means that **Text** is the input to tool T, Text∈C and T∈T. This definition is applied by ***IIE*** to check the run time relations of tools. For example, the **RB** is used to review a **resource**. That is a **resource** is the input to the **RB**. Therefore, it can be denoted as **I(resource, RB)**.

*Definition 4*: **O(T, Text)** means that **Text** is the output of tool **T**, Text∈C and T∈T. For example, the **RE** is used to edit and save a resource. That is a resource is the output of the **RE**. Therefore, it can be denoted as **O(RE, resource)**.

*Definition 5*: **D(T1, T2)** means that tool **T1** and **T2** have some dependencies, **T1**,**T2**∈T. That is there are something that are shared between tool **T1** and **T2**. If the sharable thing is changed in tool **T1**, it may influence the tool **T2**. For example, a resource is sharable by a **RE** and a **RB**. After editing by **RE**, the resource may influence the review of **RB**. In this way, it can be denoted as **D(RE, RB)**.

***Definition 6***: **Tri(T2, T1, S)** means that tool T2 can be triggered by tool **T1** after the service S is completed in **T1**, **T1,T2**$\in$**T**. That is if the service S is provided by tool T1 and listened by tool T2, tool T2 can be triggered by tool T1. For example, **P("Save digital resource", RE)** and **L("Save digital resource", RB)** then the **RB** can be triggered by the **RE** after **RE** has finished the service **"Save digital resource"**. Therefore, it can be denoted as **Tri(RB, RE, "Save digital resource")**.

***Definition 7***: **IT(T1, T2)** means that tool T1 and T2 are well-integrated, **T1, T2**$\in$**T**. That is tool **T1** and **T2** are integrated tightly if they can be triggered by each other. For example, **Tri(RB, RE, "Save digital resource")** and **Tri(RE,RB, "Resource search & not enough")** then the **RE** and **RB** are well integrated. Therefore, it can be denoted as **IT(RE, RB)**.

From the above definitions, the Integration Inference Rules(***IIR***) can be summarized as:

***Integration Inference Rule(IIR):***

***Rule 1***: **∃ T1, T2, S, Text**

**O(T1, Text) ∧ I(Text, T2) ∧ P(S, T2) → D(T1, T2)**

***Rule 2***: **∃ T1, T2, S**

**P(S, T1) ∧ L(S, T2) → Tri(T2, T1, S)**

***Rule 3***: **∃ T1, T2, T3, S1, S2**

**Tri(T3, T2, S1) ∧ Tri(T2, T1, S2) → Tri(T3, T1, S2)**

***Rule 4***: **∃ T1, T2, S1, S2**

**D(T1, T2) ∧ Tri(T2, T1, S1) ∧ Tri(T1, T2, S2) → IT(T1, T2)**

**Rule 1** describes the run time relations of tools. That is when tool **T1** outputs the **Text** which becomes the input of tool **T2** and the input **Text** should be processed by the service S of tool **T2**. It means that **T1** and **T2** have data dependency through the **Text**. This rule is applied by ***IIE*** in run time. For example, **O(RE, resource) ∧ I(resource, RB) ∧ P("Resource display", RB) → D(RE, RB)**.

**Rule 2** demonstrates the relation when tools are registered in ***TIP***. That is when a service S is provided by tool **T1**, and listened by tool **T2**. It means that tool **T2** can be triggered by tool **T1** after **T1** has finished the service S. This implies that the tool **T2** can be triggered to the running state if it is in not running or background running state. For example, **P("Resource search & not enough", RB) ∧ L("Resource search & not enough", RE) → Tri(RE, RB, "Resource search & not enough")**

**Rule 3** is applying the transitive property of **Rule 2**. For example,

**Tri(RB, RE, "Save digital resource") ∧ Tri(RE, PD, "Not enough resource input") → Tri(RB, PD, "Not enough resource input")**

**Rule 4** shows the integration status of tools. That is tools **T1**, **T2** are dependent and integrated tightly, only if these two tools can be triggered by each other, and vice versa. This rule is used to check the integration status of all tools in ***TIP***. For example, **D(RE, RB) ∧ Tri(RB, RE, "Save digital resource") ∧ Tri(RE, RB, "Resource search & not enough") → IT(RE, RB)**

The above ***IIR*** are the basic Inference Rules applied by ***IIE*** in run time. The ***IIR*** can be extended by using the above ***Definitions*** to add new ***IIR***. According to the above ***Definitions***, the phenomena of **RE**, **RB**, **PD** described in Fig. 7 can be denoted in set theory as followed:

**Resource Editor(re) --**

Let $S_{re}$ be the services provided by **Resource Editor** and denoted as $S_{re}$ = {"Load digital resource", "Save digital resource", "Resource editing"}, then **P** : $S_{re}$ → **re**.

Let $L_{re}$ be the services listened by **Resource Editor** and denoted as $L_{re}$ = {"Resource search & not enough", "Not enough resource input"}, then **L** : $L_{re}$ → **re**.

**Resource Browser(rb) --**

Let $S_{rb}$ be the services provided by **Resource Browser** and denoted as $S_{rb}$={"Resource search & not enough","Resource display"}, then **P** : $S_{rb}$ → **rb**.

Let $L_{rb}$ be the service listened by **Resource Browser** and denoted as $L_{rb}$ = {"Resource layout adjustment", "Resource schedule adjustment", "Save digital resource"}, then **L** : $L_{rb}$ → **rb**.

**Presentation Designer(pd) --**

Let $S_{pd}$ be the services provided by **Presentation Designer** and denoted as $S_{pd}$={"Resource layout adjustment", "Resource schedule adjustment", "Not enough resource input", "Presentation OK"}, then **P** : $S_{pd}$ → **pd**.

Let $L_{pd}$ be the service listened by **Presentation Designer** and denoted as $L_{pd}$={"Resource display", "Save digital resource"}, then $L : L_{pd} \rightarrow pd$.

When a tool is intended to be integrated into ***TIP***, the devveloper should use the ***CII*** to register the tool in ***TIP***. At that moment, the ***IIE*** is triggered by ***CID*** to apply the suitable ***IIR*** in the ***Repository*** to produce the appropriate tool knowledge.

The tools integrated in the ***TIP*** as described in Fig. 7 may apply ***IIE*** to deduce the tool knowledge. These tool knowledge can be drawn as a graph which is called ***Tool Dependency Graph***(TDG) and shown as Fig 8. In **TDG**, the solid line shows the direct triggering relation among tools. While the dashed line is produced through the indirect triggering of tools. It is obvious that Fig. 8 is the automatic steps of Fig. 6.

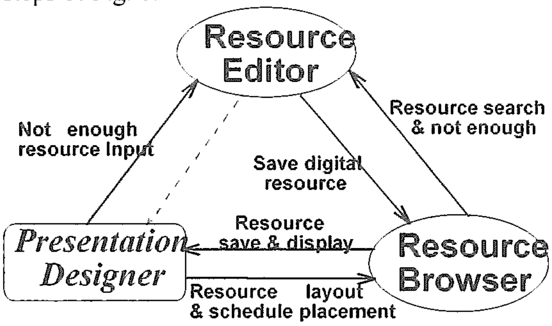


**Fig. 8 The Tool Dependency Graph of *TIP***

## 4.3 The Compared Items and Results

Some compared components are proposed to assess the ***TIP*** with the existed products and standards[4]. The existed products are HP's SoftBench(**HP**), SUN's ToolTalk(**SUN**) and Emeraude's PCTE(**PCTE**). Table 1 lists the compared results of ***TIP*** and the existed products. There are five compared components in the table. The first four compared components have been discussed in the ***vendor independent*** approach of section 2. The fifth compared component is the **Tool Integration Knowledge(TIK)** which is to compare whether the product can deduce the tool knowledge dynamically. Table 2 is the compared result of ***TIP*** and the existed integration standards such as **CORBA**, **PCTE**. From the result of Table 1, and 2, it shows that the ***TIP*** is better than the existed integration products and standards.

In addition, Table 3 is to compare the integration activity of ***TIP*** with the **CORBA** and **PCTE**. There are three components in table 3 which are **Registration(Regis)**, **Encapsulation(Encap)**, and **Interoperability(IntOpe)**. The **Regis** is to compare the provided registration method is **Dynamic** or **Static Registration(DR or SR)** for tools. The **Encap** is to compare whether the integration interfaces for integrating tools is implicit or explicit. The **IntOpe** is to compare whether the integration environment for tools is provided in different Operating Systems.

**Table 1. Compared with Existed Products**

| Item | HP | SUN | PCTE | *TIP* |
|---|---|---|---|---|
| **PI** | Provided | Provided | Provided | Provided |
| **PRI** | Provided | Provided | Provided | Provided |
| **CI** | Provided | Provided | Provided | Provided |
| **DI** | None | None | Provided | Provided |
| **TIK** | None | None | None | Provided |

**Table 2. Compared with the Existed Standards**

| Item | CORBA | PCTE | *TIP* |
|---|---|---|---|
| **PI** | Provided | Provided | Provided |
| **PRI** | Provided | Provided | Provided |
| **CI** | Provided | Provided | Provided |
| **DI** | Provided | Provided | Provided |
| **TIK** | None | None | Provided |

**Table 3. The Integration Activity Comparison**

| Item | CORBA | PCTE | *TIP* |
|---|---|---|---|
| **Regis** | DR | DR | SR and DR |
| **Encap** | Implicit | Explicit | Explicit |
| **IntOpe** | UNIX | UNIX | DOS&UNIX |

From the above comparisons, the ***TIP*** can provide a better flexibility of selecting tools, developed by different companies, to be integrated under UNIX and MS-Windows system.

## 5. Conclusion and Continuing Research

The architecture(***TIP***) discussed in this paper is to provide an environment to integrated tools in ***TIP***. In this manner, tools can compensate the drawbacks for each other. This architecture also provides the new idea of the ***IIE*** which applies suitable ***IIR*** to deduce the tool integration knowledge dynamically. The ***TIP*** is a tool integration architecture which can be applied not only limited to integrate multimedia tools but also the other fields such as the CAD, CAE, CASE ... etc. With ***TIP***, the time required to develop a new tool or modify an existed tool for tool integration can be shorten.

Thus, for continuing research, the verification of interoperability of ***TIP*** will be done. That is to integrate tools distributed among the same or different operating systems such as UNIX or OS/2 operating environment.

## References

[1]. Alan W. Brown, "Control Integration through message-passing in a software development environment", Software Engineering Journal, May 1993, PP. 121-131.

[2]. Cagan, M.R., "HP SoftBench: An architecture for a new generation of software tools", Hewlett-Packard Journal, 41, 3, June 1990, PP. 36 - 47.

[3]. CASE Interoperability Alliance and CASE Communique, "Proposed Messaging Architecture -- A Joint Proposal", December 1992, PP. 1 - 12.

[4]. Chi-Ming Chung, Wei-Chuan Lin, "Tool Cooperation in an Integration Environment by Message-Passing Mechanism", IEEE COMPSAC94, November 1994, pp. 451-456

[5]. David Sharon, "Tools That Bind: Creating Integrated Environments", IEEE Software, March 1995, pp. 76 - 85

[6]. Donald Firesmith, "An expanded View of Messages", Journal of Object-Oriented Programming, July-August 1993, pp. 51-52

[7]. Donovan Hsieh, Fred M. Gilham, Jr., "A Novel Approach Toward Object-Oriented Knowledge-Based Software Interaction within a CASE Framework", Journal of Object-Oriented Programming, September 1994, pp. 25-31.

[8]. EIA,"CDIF: organization and procedure manual", Report Number EIA/PN- 239, January 1990.

[9]. European Computer Manufacturers Association, "A reference model for computer-assisted software engineering environments", ECMA Report Number TR/55, V2, December 1991.

[10]. Frankel, R., "Introduction to the ToolTalk Service", Sun Microsystems Inc., Mountain View, California,1991.

[11]. Hal Dean, "Object-Oriented design using message flow decomposition", Journal of Object-Oriented Programming, May 1991, pp. 21-31.

[12]. Jeffrey D. Clark, "WINDOWS Programmer's Guide to OLE/DDE", Prentice Hall, 1990, PP. 4 - 239.

[13]. Ian Thomas, "Definitions of Tool Integration for Environment", IEEE Software, March 1992, pp 29 - 35

[14]. John E. Arnold, "Control Integration: A Briefly Annotated Bibliography", ACM SIGSOFT Software Engineering Notes, vol. 20, no. 2, 1995, pp. 62 - 68

[15]. Jonny Wong, "Synchronization in Specification-based MultiMedia Presentations", Software-Practice and Experience, Vol 26(1), Jan. 1996, pp. 71-81.

[16]. Mary E.S. Loomis, "Object Database -- Integration for PCTE", Journal of Object-Oriented Programming, May 1992, pp. 53-56.

[17]. Minder Chen, "A Framework for Integrated CASE", IEEE Software, March 1992, pp. 18 - 22

[18]. Petra Hoepner, "Presentation Scheduling of MultiMedia Objects and Its Impact on Network and Operating System support", 2nd International Workshop Heidelberg, Germany, Nov. 1991, pp. 132 - 143

[19]. Reiss, S.P., "Interacting with the FIELD environment", Software Practice & Experience, 1990, 20(S1), PP. 189 - 195.

[20]. S. Murrell, "Formal Semantics for Rule-Based Systems", Journal of Systems Software, January 1995, no. 29, PP. 251 - 259.

[21]. SunSoft, DEC, HP, HyperDesk, NCR, Object Design Inc., "The Common Object Request Broker Architecture and Specification", December 1991, PP. 105 - 164.