A Visual Tool for Rapid Integration of Enterprise Software Applications

Inbal Tadeski[1], Eli Mordechai[2], Claudio Bartolini[3], Ruth Bergman[1], Oren Ariel[2], and Christopher Peltz[4]

[1] HP Labs, Haifa, Israel
[2] HP Software, Yahud, Israel
[3] HP Labs, Palo Alto, California, USA
[4] HP Software, Fort Collins, Colorado, USA

Abstract. Integrating software applications is a challenging, but often very necessary, activity for businesses to perform. Even when applications are designed to fit together, creating an integrated solution often requires a significant effort in terms of configuration, fine tuning or resolving deployment conflicts. This is often the case when the original applications have been designed in isolation. This paper presents a visual method allowing an application designer to quickly integrate two products, taking the output of a sequence of steps on the first product and using that as input of a sequence of steps on the second product. The tool achieves this by: (1) copying UI components from the underlying applications user interface; (2) capturing user interaction using recording technology, rather than by relying on the underlying data sources; and (3) exposing the important business transactions that the existing application enables as macros which can then be used to integrate products together.

1 Introduction

Integrating software applications is hard. Even when such applications are designed to fit together, creating an integrated solution often requires a significant effort in terms of configuration, fine tuning or resolving deployment conflicts. If the original applications had been designed in isolation and not originally intended to provide an integrated solution, then it might easily turn into a nightmare. In fact, this scenario is very common. If you want to scare off an IT executive, tell them that “it’s just a simple application integration effort”. In reality, there is no simple and cheap application integration effort. Difficult as it is, software integration application is often necessary. Due to the dynamic nature of most enterprises, many disparate software applications live in the enterprise space. The creation of new business operations, for example, as the result of mergers and acquisitions, brings about the need to use multiple applications. The enterprise staff has to juggle between them to complete new business processes. Indeed, such processes sometimes require operators to have multiple screens open at the same time and manually move data from an application to the other. So, the IT executive has to choose between an error prone and time consuming process, or an application integration effort. The visual integration tool presents a way out for the IT executive, a method for rapid, programming-free software integration.

Software applications integrations comes in various flavors, the most prevalent of which are Enterprise Application Integration (EAI) and, more recently, web 2.0 mashups. The business relevance of this problem is exemplified by that the total available market for EAI alone, defined as the use of software and computer systems architectural principles to integrate a set of enterprise computer applications, is expected to reach 2.6 billion by 2009 (Wintergreen research). From a technology point of view, what make the problem hard are issues with data representation, data semantics, connector semantics, error control, location and non-functional requirements.

In this paper, we introduce a novel approach to software application integration, and a visual tool that embodies it, called **visual product integrator**. The main guiding principle for our approach is that software application integration should be made *faster* and *cheaper*. Our approach aims to make the *integration process* better, since our tool will enable quick proof-of-concept integrations that can go from inception to use in a very short turn-around time. We do not expect that solutions developed using it will be as robust and as complete as full standard application integration solutions. Having said this, our approach and tool, when used in parallel with standard techniques, provide the advantages of shortening developing cycles and enabling the quick and cheap creation of successive throw-away versions of the final integrated solution. Last but not least, our approach demands very little in terms of knowledge about the internal data representation and semantics of the original applications.

In a nutshell, the visual product integrator allows an application designer to quickly realize use cases for integrations of two products that can generally be defined as taking the output of a sequence of steps on the first product and using that as input of a sequence of steps on the second product. Visual product integrator achieves this by

– copying UI components from the user interface of the underlying applications.
– capturing user interaction using recording technology, rather than by relying on underlying data sources or on programming.
– exposing the important business transactions that the existing application enables as *macros*.

We refer to a recorded sequence of interactions as a macro. Once macros of one or more existing applications have been created they may be re-used in any new application. The main contributions of our approach and visual product integrator tool are:

– blend together the roles of the designer and the user of the integrated solution.
– require no programming of scripting for creating the graphical user interface of the integrated solution.

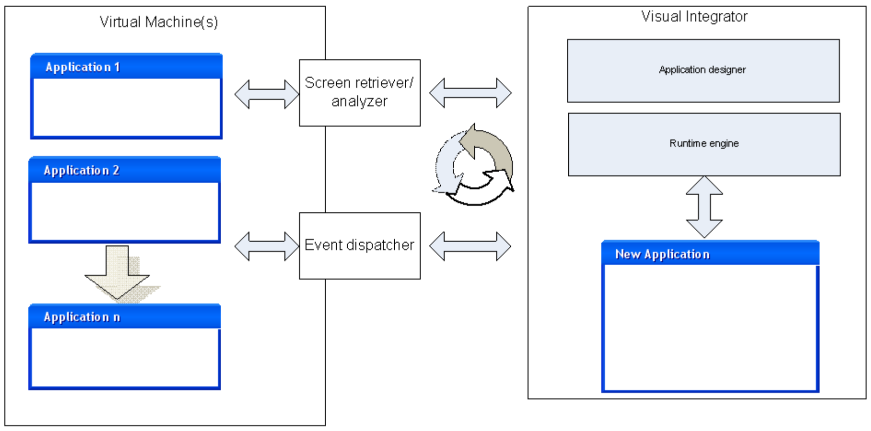


Fig. 1. Architecture Diagram for the Visual Product Integrator

- record the actions of the application user/designer similarly to what is done when recording a macro in Microsoft Excel - only this happens when juggling between different software applications rather than spreadsheets.
- force the designer/user to think about business transactions rather than specific business objects: visual product integrator captures the sequence of steps of a business process and its interactions with all the related business objects.

2 Architecture

The visual product integrator has two main functionalities. First, it allows to user to design a new application with a new user interface. Second, it provides a runtime environment for the new application. The architecture of the visual product integrator, shown in Figure 1, supports both functionalities.

At design time, the visual integrator behaves like a WYSIWYG Graphical User Interface builder. It has the usual GUI builder capabilities for placing UI components into the application interface, and assigning actions to these UI components. The visual integrator goes beyond standard GUI builders, because it can copy UI components from an existing application. To copy UI components the user is prompted to select an application, which will be started within the visual integrator. The Application Screen Retriever/Analyzer module of the visual product integrator initiates and displays the application. It also connects to the object hierarchy and captures events. The user can then select the UI components he wishes to copy into the new application: the visual integrator has the ability to duplicate the selected components in the new application. Copied components are tagged, so that at run time they can be synchronized with the original application. The user can continue to copy objects from any number of applications. An additional enhancement in the visual product integrator is that the actions assigned to UI components may include *macros*. At design time, macros are recorded using the record/replay engine. It is also possible to import macros that were recorded or manually written elsewhere.

At run time, the visual product integrator must do all of the following

1. Run the new application
2. Refresh copied UI components from the original application
3. Run macros on original application

Referring to the architecture diagram in Figure 1, the Runtime Engine is responsible for running the new application. The critical new piece of this technology is the Application Event Dispatcher, which handles the events of the new application that refer to any of the original application. It dispatches events from copied UI components of the new application to the original application. Similarly, it refreshes the state of copied UI components based on the state of the respective component of the original application. The Application Event Dispatcher also handles macros using the record/replay engine.

Notice that we describe the visual product integrator independently of the technology used for the underlying applications. Our architecture is rich enough to enable combining Windows applications with HTML applications, Java applications, and so on. All is required is that the Application Screen Retriever/ Analyzer be able to manipulate and analyze the object hierarchy of the application. It must be able to convert UI components from the original technology to the technology used for the new application. For example, if the technology for the new application is HTML and the underlying application is .NET, the Screen Retriever/Analyzer must recognize that a selected UI component is, e.g., a .NET button, so that it can tell the Application Designer module to create an HTML button. Likewise, the Application Event Dispatcher must be able to replicate events on all the technologies of the underlying applications and query the objects for their current state. These capabilities require a deep model of the object hierarchy for each supported technology, similar to what is used for automated software testing, e.g., in HP Quick Test Pro [3], for example. By leveraging this technology, the visual integrator can support standard Windows, Visual Basic, ActiveX controls, Java, Oracle, Web Services, PowerBuilder, Delphi, Stingray, Web (HTML) and .NET.

The integrator is a GUI builder application, among other things. As such it builds new applications in some particular technology. This technology can be Java Swing, .NET, HTML or any other GUI technology. It is important to note that every application built by the visual product integrator will use this same GUI technology.

To achieve a good runtime experience, we want to present a single application to the user. Although there are several underlying applications and all these applications are running while the new application runs, we want to hide the presence of these applications from the user. In the architecture, therefore, the original applications run using a remote protocol, either on a remote machine or on a Virtual Machine (VM), rather than on the local machine. At design time, the Screen Retriever/Analyzer module displays the underlying application to the user and captures the UI objects that the user selects. At run time, the Application Event Dispatcher passes events to and from the VM, and refreshes the components of the new application.

3 Implementation

The prototype implementation of the visual product integrator utilizes the architecture described in the previous section. We made several simplifying assumptions that enable us to rapidly prototype and demonstrate feasibility of the visual product integrator. The technology used to develop the new application is HTML. In our first embodiment of the visual product integrator, we are also restricting the underlying applications to a single technology, again HTML. With this restriction, the Application Screen Retriever/Analyzer and the Applications Event Dispatcher need only have a model of a single technology. Much of the functionality of the Applications Event Dispatcher is the macro record and replay. This capability is available in the software testing application Quick Test Pro (QTP) [3]. For the prototype we use QTP's record/replay engine to dispatch events on the underlying applications. QTP itself runs on the VM in order to give the user the expected interaction with the application.

Figures 2(a) and 2(b) show screen shots of the Visual Application Integrator prototype. To design a new application, the user does the following:

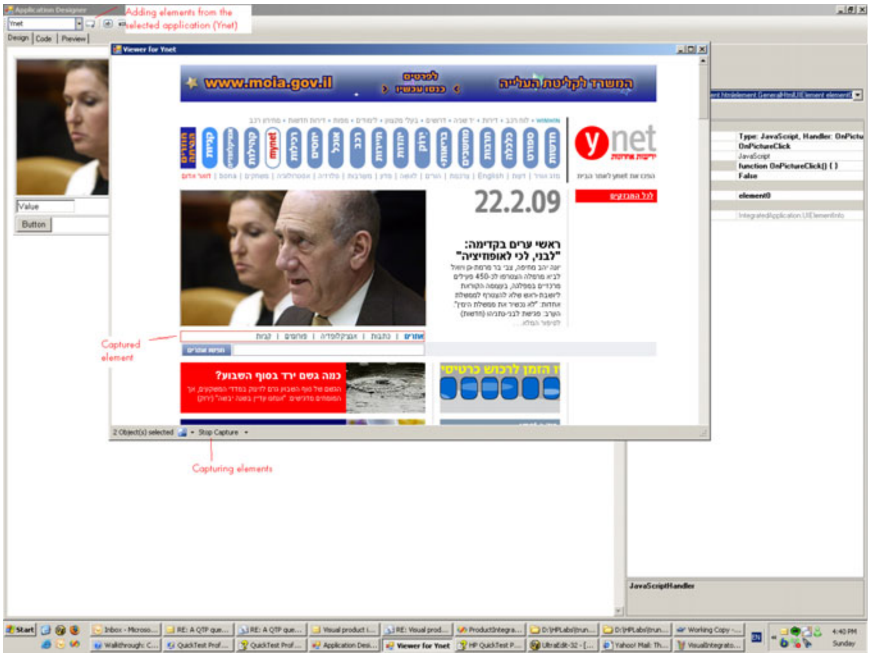
Select Application. Figure 2(a) shows the design time view in the integrator. The user chooses one or more applications, which the integrator initiates and displays.

Copy UI Components. The user can select UI objects for duplication in the new application. Selected objects appear in the design view of the integrator, as shown in Figure 2(b).

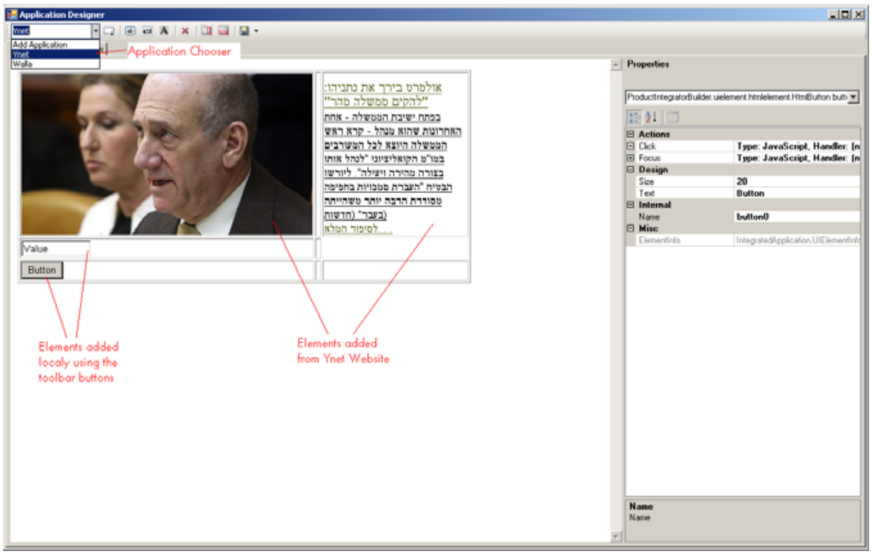
Add New UI Components. The user may add additional UI components from the toolbar.

Set Actions. For each object the user assigns an action. Figure 2(b) demonstrates the set of possible actions.

1. If the element was copied from one of the underlying applications, the user selects the"keep original behavior" mode, then the object will execute the original action in the original application.
2. New objects may be assigned actions by writing a java script function. These functions will be executed locally.
3. Another type of action for a new object is a "workflow action". The workflow is set up by connecting the action with a QTP script. The QTP script will run on the original application in the VM. In the current prototype we do not enable recording the QTP script from the integrator. Instead, we assume that the QTP script is pre-recorded. This limitation is related to the implementation of the record/replay engine in QTP. Future implementations using a record/replay engine built specifically for the integrator will not have this limitation.

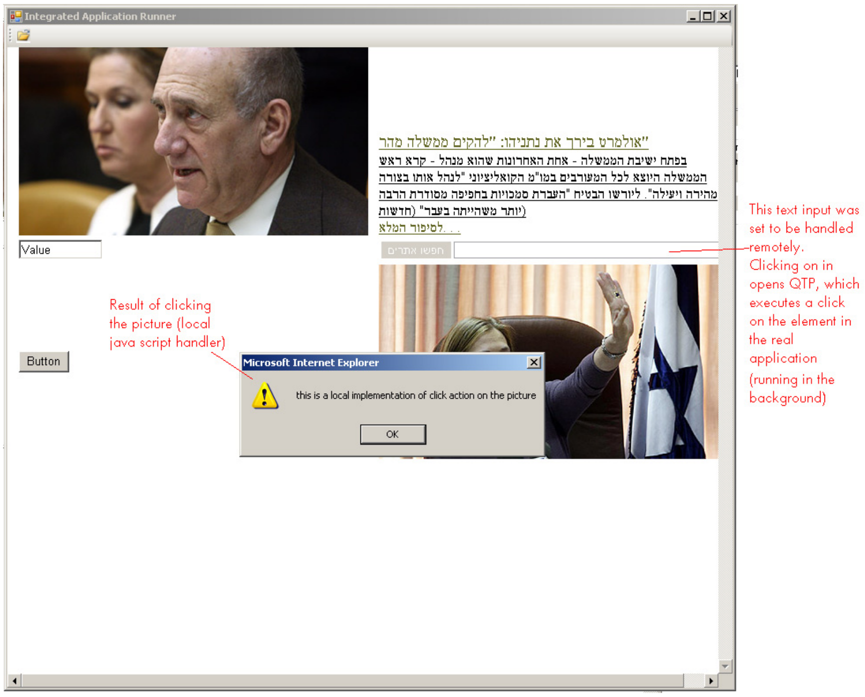


(a) A view of capturing elements from the underlying application



(b) The user can arrange UI elements, and assign actions to each element

Fig. 2. Visual Product Integrator screen shots depicting design capability



"אולמרט בירך את נתניהו: "להקים ממשלה מהר

בפתח ישיבת הממשלה - אחת האחרונות שהוא מנהל - קרא ראש הממשלה היוצא לכל המעורבים במו"מ הקואליציוני "לנהל אותו בצורה מהירה ויעילה". ליורשו הבטיח "העברת סמכויות בחפיפה מסודרת הרבה (יותר משהייתה בעבר" (חדשות

לסיפור המלא. . .

this is a local implementation of click action on the picture

Result of clicking the picture (local java script handler)

This text input was set to be handled remotely. Clicking on in opens QTP, which executes a click on the element in the real application (running in the background)

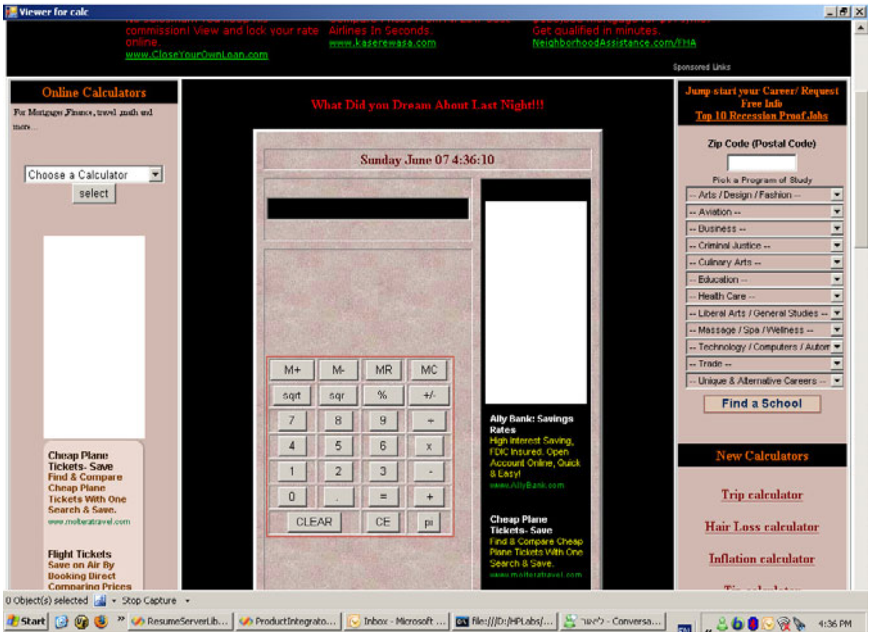
Fig. 3. Visual Product Integrator screen shot. A view of the application at run time.

Figure 3 shows the run time view in the integrator. The integrator is now running a HTML application in a browser object. But this application must run inside the integrator because of the connection with the underlying applications. First, the integrator refreshes copied objects, as necessary. Second, actions on copied UI objects and QTP scripts are executed on the original applications. The results of these actions are shown in the new application. The integrator uses QTP to start and drive the application. QTP gets its instruction via its remote automation API, thus hiding the presence of these applications from the user. Screen data is captured and reflected in the host machine via new remote API.

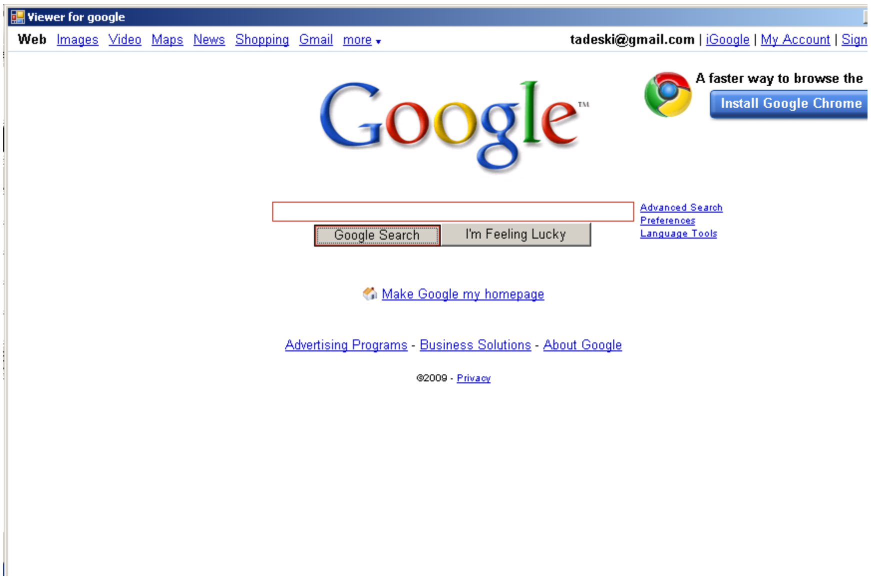
4 Validation

In order to validate our tool, we used it to build a **Search-Calculator** application. This application uses functionality from two Web applications, a calculator application [1] and the Google search application [2].

To build the new application the user first opens each application in a viewer. Figure 4 shows each application in the viewer. Next the user selects UI elements for copying. Figure 4(a) shows the calculator application, in which all the elements making up the calculator interface have been selected for copying to the

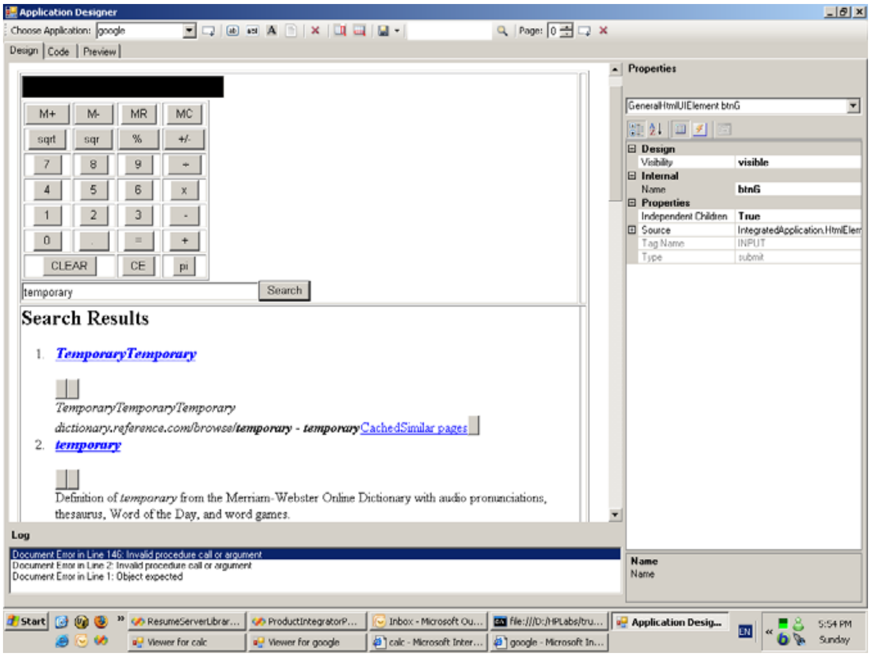


(a) Object capture from the calculator application


Viewer for google
Web Images Video Maps News Shopping Gmail more
tadeski@gmail.com | iGoogle | My Account | Sign
Google
A faster way to browse the
Install Google Chrome
Google Search
I'm Feeling Lucky
Advanced Search
Preferences
Language Tools
Make Google my homepage
Advertising Programs - Business Solutions - About Google
©2009 - Privacy

(b) Object capture from the Google search application

Fig. 4. Building the Search-Calculator application. A view of the underlying applications inside the viewer.



temporary

Search Results

1. *TemporaryTemporary*

TemporaryTemporaryTemporary
*dictionary.reference.com/browse/**temporary** - **temporary***CachedSimilar pages

2. *temporary*

Definition of *temporary* from the Merriam-Webster Online Dictionary with audio pronunciations, thesaurus, Word of the Day, and word games.

GeneralHtmlUIElement btnG

Design	
Visibility	visible
Internal	
Name	btnG
Properties	
Independent Children	True
Source	IntegratedApplication.HtmlElem
Tag Name	INPUT
Type	submit

Name
Name

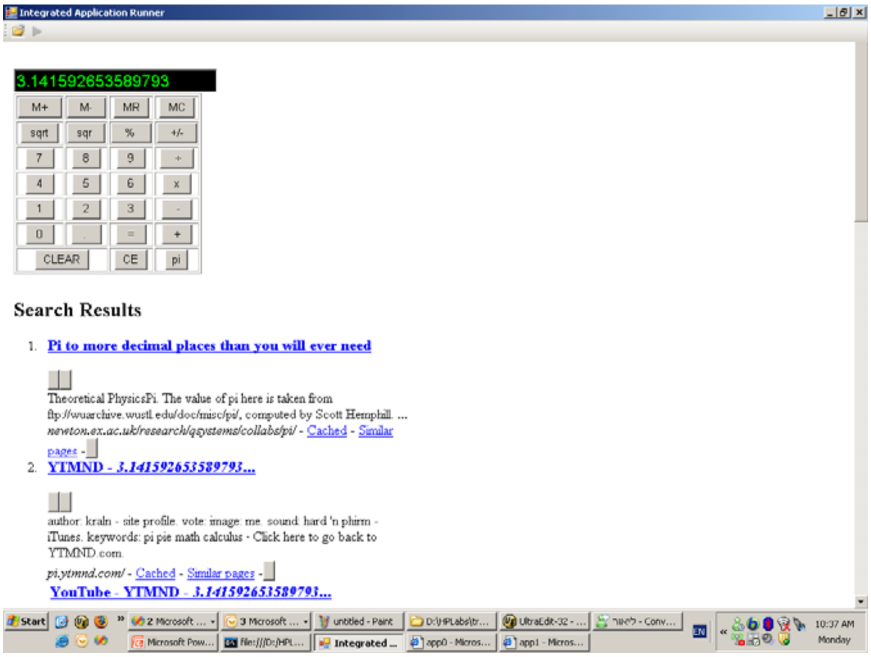
```
Document Error in Line 146: Invalid procedure call or argument
Document Error in Line 2: Invalid procedure call or argument
Document Error in Line 1: Object expected
```

Fig. 5. Building the Search-Calculator application. The user arranges UI elements and sets actions in the designer.

new application's UI. Figure 4(b) shows the selection of the elements on the Google page that are needed for search, i.e., the edit box and the search button.

The next step of building the new application includes arranging UI elements, and assigning actions to each element. The designer supports these activites, as shown in Figure 5. For the **Search-Calculator** application, the actions of the calculator components are set to remote, meaning that actions on these buttons will be passed through to the original calculator application. Likewise, the calculator result text box is remote, so the value calculated in the original application will be automatically updated in the new application. Similarly, remote actions are set on the elements copied from the Google application. Finally, the workflow action of searching for the calculator result is set using a QTP script. That completes the application.

At run time, the **Search-Calculator** application will search the Web for the value resulting from any computation. Figure 6 demonstrates this behavior. The user pressed the PI button on the new application's interface. The action was passed to the calculator application, which displays the value 3.14159. The new application refreshes the value on the calculator display due to the change in the underlying application. Based on the workflow set up earlier, the value 3.14159 is copied to the search edit box and the search is started. When the search results are available in the Google search application, the new application interface is refreshed to show the results.



Search Results

1. Pi to more decimal places than you will ever need

   Theoretical PhysicsPi. The value of pi here is taken from ftp://wuarchive.wustl.edu/doc/misc/pi/, computed by Scott Hemphill. ...
   newton.ex.ac.uk/research/qsystems/collabs/pi/ - Cached - Similar pages -
2. YTMND - *3.141592653589793...*

   author: kraln - site profile. vote: image: me. sound: hard 'n phirm - iTunes. keywords: pi pie math calculus - Click here to go back to YTMND.com.
   pi.ytmnd.com/ - Cached - Similar pages -

   YouTube - YTMND - *3.141592653589793...*

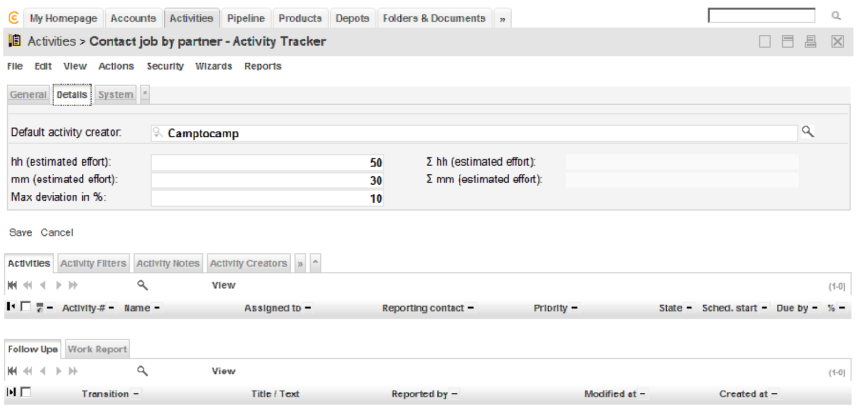
Fig. 6. Visual Product Integrator screen shot. A view of the **search-calculator** application at run time.

5 An Enterprise Application Integration Use Case

The benefit of this tool for enterprise application integration is illustrated by the following a common scenario. In this scenario, an operator has to juggle between two applications to carry out their daily task. The operator opens the Customer Relationship Management (CRM) applications and finds a list of activities. Among them, he sees a new "Contract job by partner" activity. He clicks on the *detail* tab of the activity. The operator then can navigate through the details of the activity, shown in Figure 7(a). This screen shows the activity creator *camptocamp*, a partner company.

At this point - as is often the case with enterprise applications - the operator will need to pull up an Enterprise Resource Planning (ERP) application, to retrieve collateral information about the partner, in the partner events panel. He will have to enter in the basic search form (possibly copy-and-paste) the name of the partner as it appears in the CRM application. On obtaining the search results, the operator has to add a new event for this partner and fill the details manually with information based on the activity he found in the CRM application (Figure 7(b)).

Consider using the visual product integrator to expose the operator to a single interface, and to automate this error-prone, manual task. To build the new application, the application designer selects and copies a UI element from the first



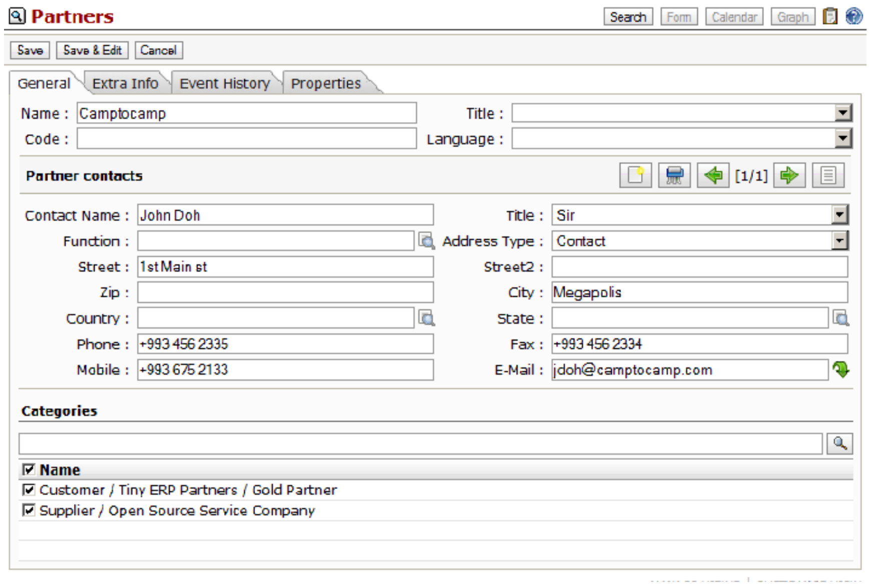
Activities > Contact job by partner - Activity Tracker

Default activity creator:	Camptocamp		
hh (estimated effort):	50	Σ hh (estimated effort):	
mm (estimated effort):	30	Σ mm (estimated effort):	
Max deviation in %:	10		

#	Activity-#	Name	Assigned to	Reporting contact	Priority	State	Sched. start	Due by	%

Transition	Title / Text	Reported by	Modified at	Created at

(a) A Customer Relationship Management (CRM) application. View of a new "Contract job by partner" activity.



Name :	Camptocamp	Title :	
Code :		Language :	

Partner contacts

Contact Name :	John Doh	Title :	Sir
Function :		Address Type :	Contact
Street :	1st Main st	Street2 :	
Zip :		City :	Megapolis
Country :		State :	
Phone :	+993 456 2335	Fax :	+993 456 2334
Mobile :	+993 675 2133	E-Mail :	jdoh@camptocamp.com

Categories

☑ Name
☑ Customer / Tiny ERP Partners / Gold Partner
☑ Supplier / Open Source Service Company

(b) An Enterprise Resource Planning (ERP) application. Entering new partner activity by copying from the CRM application.

Fig. 7. An enterprise application integration example

original application, i.e., the CRM application shown in Figure 7(a). He then adds a new action button, labeled "update ERP". At this point, the tool makes use of recording technology to capture the business flow that the operators has to execute, and attaches a description of that to the button. The tool records the sequence of copy and paste operations between the selected fields from the first original application (CRM) to the second original application (ERP). As the designer performs a function on the values from the selected fields in the first original application to fill in the value of the selected field in the second original application, visual product integrator records that function and sets this macro as the action on the button in the new application. Thus, this button represents the business flow of creating a new event for a partner and copying the details from the activity.

6 Related Work

The Visual Product Integrator tackles the problem space covered by enterprise application integration (EAI), but aiming at making it significantly cheaper and faster, getting inspiration from UI integration concepts such as mashups and Web 2.0, which today are aimed at web application integration.

6.1 Enterprise Application Integration

EAI applications usually implement one of two patterns [14]:

Mediation: The EAI system acts as the go-between or broker between the underlying applications. Whenever an interesting event occurs in an application an integration module in the EAI system is notified. The module then propagates the changes to other relevant applications.

Federation: The EAI system acts as the front end for the underlying applications. The EAI system handles all user interaction. It exposes only the relevant information and interfaces of the underlying applications to the user, and performs all the user's actions in the underlying applications.

Most EAI solutions, whether they implement the mediation or federation approach, focus on middleware technology [15], including: Message-oriented middleware (MOM) and remote procedure calls (RPCs); Distributed objects, such as at CORBA and COM; Java middleware standards; and Message brokers The visual product integrator, by contrast, uses the federation pattern.

A long list of EAI solutions may be found in [18]. We mention some leading solutions in this area. Microsoft BizTalk Server [16] is used to connect systems both inside and across organizations. This includes exchanging data and orchestrating business processes which require multiple systems.

Microsoft Host Integration Server allows enterprise organizations to integrate their mission-critical host applications and data with new solutions developed using the Microsoft Windows platform. All these middleware solutions require the application integrator to go deep into the original application's source code and write wrappers that bring the original application into the fold of the new technology.

In contrast to most EAI solutions, our approach eliminates the need to access the application's source code, substituting that with a representation of the user-interaction that does not depend on the original applications information model.

6.2 Integration at the Presentation Layer (Web Applications)

The web engineering community has so far typically focused on model-driven design approaches. Among the most notable and advanced model-driven web engineering tools we find, for instance, WebRatio [11] and VisualWade [10]. However these approaches tend to be heavy on modeling methods, thereby being more similar to traditional EAI in spirit than the approach we take here.

A very good account of the of the problem of integration at the presentation layer for web applications is found in [13]. On thoroughly reviewing academic literature, the authors drew the conclusion and concluded that there are no real UI composition approaches readily usable in end-users composition environments. There are some proprietary Desktop UI component technologies such as .NET CAB [9] or Eclipse RCP [8], and their being proprietary in nature limits their possible reuse for software product integration in general. On the other hand, browser plug-ins such as Java applets, Microsoft Silverlight, or Macromedia Flash are easily embedded into HTML pages, but provide limited and cumbersome interoperability through ad-hoc plumbing.

Mashup approaches, based on Web 2.0 and SOA technologies have been gaining broad acceptance and adoption [13]. Existing mashup platforms typically provide easy-to-use graphical user interfaces and extensible sets of components for development composite applications. Yahoo! Pipes [12] integrate data services through RSS or Atom feeds, and provide a simple data-flow composition language. UI integration is not supported. Microsoft Popfly [7] uses a similar approach and does provide a graphical user interface for the composition of data, application, and UI components, however does not support operation composition through different components. IBM Lotus Mashups [4] provides a wiki-based (collaborative) mechanism to glue together JavaScript or PHP-based widgets. Intel MashMaker [5] takes a different approach to mashup of web application in that it makes "mashup creation part of the normal browsing process". As a user navigates the web, MashMaker gives them suggestions for useful mashups to be applied to the page that they are looking at. As a result, mashups are composed "on-the-go", but the main use case for MashMaker is still "Pipes"-like. It is not oriented to producing a new stand-alone application like our visual integrator. JackBe Presto [6] has an approach similar to Pipes for data mashups and provides mashlets, or UI widgets used for visualizing the output. Throughout all these example, we observe that mashup development is a process requiring advanced programming skills.

6.3 Other Hybrid Approaches

One closely related solution for product integration is the OpenSpan platform [17]. This platform may be used to build new composite applications by integrating any legacy applications and/or business workflows in a rapid manner. By dynamically hooking into applications at run-time,

OpenSpan can create integration points with an application without requiring modification or access to the application's source code, APIs or connectors. At design time, the user can set up the workflow of a business process, by hooking into UI objects and specifying the data flow between the UI objects. We observe two main differences between the OpenSpan platform and our Visual Product Integrator:

1. Our integrator captures the workflow or business process as macros, using record/replay technology. This technology spares the designer the manual task of setting up the flows.
2. Our integrator allows the creation of a new graphical user interface. The user of the new application only sees this interface. While the underlying applications are running, they are not visible. Thus, the user experience is more satisfying.

7 Conclusions

We presented a visual tool for rapid integration of software products to create a new application. The main advantages of this tool over prior solutions are

– It goes beyond simple juxtaposition and syndication of data feeds well into the realm of visual application design.
– It eliminates the need to access the application's source code, substituting that with a representation of the user-interaction that does not depend on the original application's information model.
– It captures the workflow or business process as macros, using record/replay technology. This technology spares the designer the manual task of setting up the flows.
– It allows the creation of a new graphical user interface. The user of the new application only sees this interface. While the underlying applications are running, they are not visible. Thus, the user experience is more satisfying.

The current prototype implementation has several limitations. The most severe limitation is that it uses Quick Test Pro (QTP) as the record/replay engine. The advantage of QTP is that it is a mature product. The disadvantage is that it is cumbersome. We make it transparent to the user by running it on a VM. In future development, we would replace QTP with a lighter weight record/replay engine. A second limitation of the prototype is that, at present, it is restricted to HTML applications. Future work will add support for additional GUI technologies, such as .NET and Java. Longer term work is aimed to eliminate the need to model specific GUI technologies by modeling the object hierarchy directly from the application's visual interface, i.e., the screen images.

References

1. CalculateForFree, `http://www.calculateforfree.com/`
2. Google search, `http://www.google.com/`

3. HP Quick Test Pro, https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-241352_4000_100__
4. IBM Lotus Mashups, http://www-01.ibm.com/software/info/mashup-center/
5. Intel Mash Maker, http://mashmaker.intel.com/web/
6. JackBe Presto, http://www.jackbe.com/
7. Microsoft popfly, http://en.wikipedia.org/wiki/Microsoft_Popfly
8. Rich Client Platform. Technical report, The Eclipse Foundation
9. Smart Client - Composite UI Application Block. Technical report, Microsoft Corporation
10. VisualWade, http://www.visualwade.com/
11. WebRatio, http://www.webratio.com
12. Yahoo! Pipes, http://pipes.yahoo.com/pipes/
13. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding ui integration: A survey of problems, technologies, and opportunities. IEEE Internet Computing 11, 59–66 (2007)
14. Hohpe, G., Bobby, W.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, Reading (2003)
15. Linthicum, D.S.: Enterprise Application Integration. Addison-Wesley, Reading (2000)
16. Microsoft Biztalk Server, http://www.microsoft.com/biztalk/en/us/default.aspx
17. OpenSpan, http://www.openspan.com/index.php/software_platform.html
18. Wikipedia, http://en.wikipedia.org/wiki/Enterprise_application_integration