IT 08 026

Examensarbete 15 hp
Juni 2008

Integration for dummies

Searching for a suitable interface for e-learning

Jonas Jakobsson

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Integration for dummies

Jonas Jakobsson

The integration competence center of Sandvik Systems Development needed a manual for their system integration solution called Total Business Integration. The point of the manual was to make easy the understanding for somebody inexperienced with the process. Their system integration solution contained a few self made integration components, which new employees and consultants should be able to quickly understand by reading the manual. The work was divided into gathering material for the manual, and making the interface for the manual. The interface was a web browser interface made in Flex with a backend in web services and an Mssql database. The interface was designed in iterations using a user centered approach. The design choices were based both on established human computer interaction and general design guidelines for computer interfaces, as well as user centered methods like PACT analysis. The idea to be tested was whether dynamic diagrams were useful in an e-learning application.

Handledare: Patric Ferry
Ämnesgranskare: Iordanis Kavathatzopoulos
Examinator: Anders Jansson
IT 08 026
Sponsor: Sandvik Systems Development AB

Tryckt av: Reprocentralen ITC

Populärvetenskaplig sammanfattning

Människa-dator-interaktion (MDI) är en vetenskap som bland annat handlar om förståelse för vad som gör datorsystem lättanvända. MDI är en multidisciplinär vetenskap, vilket betyder att den spänner över många andra vetenskapliga områden. Bland andra finns psykologi, ergonomi, lingvistik och datavetenskap.

Mitt examensarbete gick ut på att göra en manual över ett antal datorsystem. Manualen skulle vara så användarvänlig som möjligt. Jag tänkte då genast på människa-dator-interaktion som en väg till att göra manualen lättanvänd. En manual behöver ju inte vara i pappersform. Den kanske kan bli ännu mer lättanvänd genom att vara något man tittar på med hjälp av en dator.

Arbetet skulle utföras på Sandvik Systems Development (SSD). SSD tillhandahåller IT-tjänster till övriga bolag inom Sandvikkoncernen, tillsammans med Sandvik Information Technology (SIT). Båda är internationella IT-företag med endast Sandvikkoncernen som kund.

De har en sammanlagd omsättning på cirka en miljard tillsammans och är skiljda åt för att de har olika inriktning. I grova drag sköter SIT underhåll och support medan SSD sköter systemutveckling.

Sandvik är ett snabbt växande företag, mycket på grund av nyförvärv av andra bolag. Det sker i nuläget i en takt av ungefär ett företag i månaden.
Alla dessa företag måste integreras i Sandvik så att man kan få positiva effekter av köpet. Exempelvis kanske man kan minska kostnaderna för IT genom att centralisera systemadministration och datasupport.
Vad Sandvik får när de köper ett bolag kan exempelvis vara lite nya datorsystem. I de fall där systemen behålls så behöver de vanligtvis integreras med redan befintliga system.

Till denna integrationsprocess kommer att man ständigt upptäcker nya sätt att utnyttja det data som finns inom organisationen, samt att man också ställer högre krav på tillgängligheten för informationen. Man behöver nya kopplingar mellan existerande datorsystem för att göra informationen så lättillgänglig som möjligt. Allt på ett säkert sätt naturligtvis.
Lättillgänglig information är mycket viktigt för ett företag. Inte minst på grund av ständigt ökande krav på Business Intelligence, som i grova drag handlar om att fatta bra beslut baserad på så bra information som möjligt om hur läget är i företaget.
Det finns andra fält som också bottnar i ett behov av stora mängder data. Ett exempel är Data Mining. Data Mining går i korthet ut på att se mönster och utläsa information ur existerande data. Kanske genom att se tendenser i kunders köpvanor som i sin tur leder till nya affärsmöjligheter för företaget.

SSD har tagit fram ett koncept för systemintegration som kallas för Total Business Integration (TBI). I TBI-konceptet ingår diverse program och system, gemensamt kallade för systemintegrationskomponenter. Bland dessa komponenter finns iBridge, Jintegrator och SanSpeak. De är egenutvecklade program på SSD. Förutom dessa program finns BizTalk som en större komponent i TBI.

Det var dessa system jag skulle göra en manual för så man lätt kunde sätta sig in i koncepten och skulle kunna gå in på de detaljer man var särskilt intresserad av.

Det finns saker som talar både för och emot att just ett datorgränssnitt skulle vara en bra form för TBI-manualen. Något som talar för är till exempel att en människa lättare tar till sig information om den är kodad på flera olika sätt. Det vill säga att man förmedlar informationen på flera sätt. Till exempel så kan man ha en text som förmedlar informationen. Sedan en bild som också förmedlar den. En dator kan ju presentera information på fler sätt än text och bild.
En människa tenderar också att ha ett bra bildminne, så visuell information är ett bra komplement till text. Även video är ett bra alternativ att för presentera information. Man bör använda sig av färger och andra tekniker när man presenterar text och framför allt inte presentera användaren med en vägg av enfärgad text. Det sistnämnda skrämmer bara iväg användaren.

En pappersmanual har som sagt i princip bara möjlighet att koda information i text och bild.
I datamedia är man endast begränsad av tillgängliga gränssnitt mellan användare och dator.
I dagsläget innebär det att man kan visa vad som helst som är visuellt, som en film eller ett interaktivt program som användaren kan använda och interagera med. Att ge information i ljudform är också det en möjlighet. Det har också nyligen blivit möjligt att presentera verkligt holografiska tredimensionella bilder för betraktaren genom ett datorgränssnitt.

En sak till som gör att ett datorgränssnitt skulle kunna vara bättre än en pappersmanual är att det har visat sig att användare lättare kommer ihåg informationen om man försöker komma ihåg den under samma förutsättningar som när man lärde sig den. Till exempel så kommer en dykare lättare ihåg information de lärt sig när de dykt om de försöker minnas informationen under ett dyk.
En användare som lär sig om TBI kan tänkas göra det i sin arbetsroll, och då är användande av dator en ganska naturlig del av inlärningen eftersom en typisk användare sitter framför datorn på sitt arbete. De kan alltså lära sig i just den omgivning som de sedan har användning för informationen i.

Till nackdelarna med ett datorgränssnitt hör främst saker som har att göra med hur vanliga datorer idag ser ut.
Man läser till exempel oftast lättare i en pappersbok än på en datorskärm. Riktigt bra skärmar finns inte på vanliga datorer idag. Man har ibland tillgång till pappersinformationen snabbare än datorinformation eftersom datorer av idag är ganska långsamma.
Det kan också vara ergonomiskt bättre att läsa en pappersmanual. När man använder en dator är man ofta begränsad av batteritid, närhet till eluttag, nätverksuttag, obekväma kontorsmöbler, eller liknande.

Vad jag ville göra var att undersöka om den designidé för användargränssnittet jag hade kommit på kunde fungera för TBI-manualen jag skulle göra. Det gjorde den inte fullt ut, och i uppsatsen går jag igenom varför.

Acknowledgements

I would like to thank Kjell Persson at Sandvik Systems Development. It was such a coincidence that I got to do this thesis work, and it was to a great extent thanks to him.

I would also like to thank my fiancée Frida Sjöqvist for giving me both motivation to write and pointers on the actual writing, as well as giving feedback during the whole thesis work.

Iordanis Kavathatzopoulos was also very helpful. He gave some hints on possible problem formulations and what literature I could use.

Thanks to Patric Ferry, my supervisor at Sandvik Systems Development.
He was very helpful in all practical stages of the thesis work and provided insight into many integration systems that I needed some help with.

Thanks to my friend Anders Johansson for spelling and grammar corrections.

Thanks also to Anders Rackner, Thomas Lundström, Niclas Gabrielli and Magnus Wiksell for putting up with my questions and interviews.
All the above share my sincerest gratitude!

Abbreviations

BI	Business Intelligence
DM	Data Mining
ESB	Enterprise Service Bus
HCI	Human Computer Interaction
ICC	Integration Competence Center
MDI	Människa-dator-interaktion
SIT	Sandvik Information Technology
SOA	Service Oriented Architecture
SSD	Sandvik Systems Development
TBI	Total Business Integration
WCF	Windows Communication Foundation
IDE	Integrated Development Environment

Table of contents

1. Introduction

The background and motivation for the thesis is presented as well as an overview of the work that was carried out at Sandvik Systems Development. Finally the structure and limitations of the work is presented.

1.1. Background

"Information wants to be free"[1] is an expression that, at least when it comes to information in corporate data systems, appears to have some bearing on reality. It may be viewed as an example of a generalization of Parkinson's so called law[2, 3].
That is, companies are struggling with increasing amounts of data and an increased need to extract as much information from it as possible[4]. New legislations require companies to store data for longer time[5]. Old computer systems must somehow be integrated with new systems and company acquisitions are also a main factor behind integration needs[4].

Sandvik AB is acquiring about one new company each month[6] and is running computer systems that have been around since the 1970´s[7]. This means that Sandvik is relying on system integration in its information infrastructure.

System integration is about connecting computer systems with each other so that the information flows as freely as possible to where it is needed[21]. There might be many positive effects of using system integration. For example it might give the company competitive advantages or lead to the discovery of new business opportunities[4].
Sandvik Systems Development (SSD) has a solution to the integration problems within Sandvik that they call TBI, which stands for Total Business Integration[21].

The TBI concept comprises a number of software systems. Each component is used in a specific manner, so as to solve a specific problem. TBI is centered on technical concepts such as XML, SOA and ESB[21].

A few of these systems were developed by SSD[21]. One of those, called iBridge, received the 2007 .NET award in Sweden[8]. A few of the integration systems were built using the .NET languages, but since Sandvik has a very broad range of platforms, there also exists other platforms for the integration solutions. One of those solutions is called Jintegrator which uses Java and may run on a few different platforms, but is mainly used on mainframe computers[13].

The practical work for this thesis, carried out at SSD, was to make a manual for the TBI concept. The manual should be as easy to learn from as possible. The work consisted of conducting a number of interviews in order to establish what each component in TBI was doing and then analyze the result and write a manual.

This thesis is about the problem of finding a good interface for that manual, and not about system integration even though that topic is interesting in its own right.

A text and picture version of the manual created during this thesis is included in the appendix and the graphical interface may be viewed on the thesis homepage[9].

1.2. Purpose and goal

The purpose with this thesis is to investigate if the interface design idea that I came up with was suitable for the task at hand, which was to make learning about the TBI concept as easy as possible for the intended users. The goal is a usable and reusable interface that may hold instructional content like the TBI manual.

1.3. Delimitations

The TBI concept is not covered in full in the manual. Only 4 components are described. They are iBridge, SanSpeak, Jintegrator and BizTalk. The level of detail for each component is limited as well. The interface was to be evaluated, but the user testing was limited to a small group of users due to resource constraints. Interface design guidelines are discussed, but I am not going to be complete in my coverage of good design practices and principles for computer interfaces.

1.4. Workflow

The work started by the conduction of interviews with employees of the Integration Competence Center (ICC) group at SSD in order to gather material for the TBI manual and discover possible implementation solutions for the manual. Along with these interviews I conducted a literature study and tried to find a problem formulation for the thesis. The problem formulation became clear to me in discussions with my reviewer at Uppsala University, Mr. Iordanis Kavathatzopoulos.
After establishing what TBI was, the interface of the manual was created and refined incrementally in three iterations.
In all the iterations the interface was evaluated by a population of users that was as close to the actual users as possible.

1.5. Disposition

The thesis begins in chapter 1 with an overview of the background and the problem that I tried to solve. In chapter 2 I proceed to the methods used in the thesis work.
In chapter 3 I describe the design process and the results of the user reviews.

In chapter 4 the final result of the interactive design is presented, and how it was compared against the goals. Chapter 5 discusses the reasons why the results were not entirely matching the goals, and what could have been done better.
And finally in chapter 6, the appendix, the TBI manual is presented.

2. Methods

Method selection is explained. Other alternatives are discussed.

2.1. Literature study

A literature study was conducted by the help of searching on Google, Yahoo, Disa and Diva[10] for literature and related work. A central resource was the course literature[11] for the A-level course in HCI at the department of information technology at Uppsala University.

An additional resource that could have been beneficial to use was to get the help of a librarian at the university library. They offer help with finding references for thesis works.
Instead I borrowed books and read research literature at the University library that sounded like they could be of use, but they often were not.

2.2. TBI concept interviews

Human to human communication is more effective in conveying information than computer to human communication due to the fact that humans may use more than the spoken language when communicating. That makes interviews a good candidate to gather information about TBI. Interviews were held with employees that had expert knowledge about the TBI component of interest, which were iBridge[12], Sanspeak[20], Jintegrator[13] and Biztalk[14]. I also interviewed my supervisor[21] about TBI.

If the interviews are explorative in nature and there is no background information beforehand one should use a qualitative instead of a quantitative approach[15]. Since I had some background information I chose a combined qualitative and quantitative approach. The interviews then had to be mostly unstructured due to my lack of understanding of the TBI concept. The interviews were mostly unstructured where the component expert explained the component and its architecture. There were however a few questions that I asked for each of the TBI components.
They were:

- What platform does the TBI component run on?
- How is the component used today?
- How should it be used?
- How should system integration ideally be done in the future?
- Is testing a part of the system, or is testing used in the integration implementations?

2.3. Initial design methodology

The requirements for the interface application for the manual was gathered from the initial thesis problem formulation and from conclusions derived from the intended types of users during interviews with the future owners of the manual, which was the ICC group at SSD.

Requirements were also based on discussions with my supervisor at SSD[21] and the head of the ICC department. A PACT analysis[11] was also conducted in order to establish application requirements.

Gathering requirements by interviews is good since misunderstandings may be immediately recognized and attended to[11]. Since the interviews were limited to only one person at a time, there was less risk that some interested party could have more to say about the design than they should have[11].

A more structured way of gathering requirements could have been used, if there had been more time. For example doing interviews with a focus on requirements and collecting user stories and then synthesize them down to use cases, as is the recommended approach in “Designing interactive systems”[11].
User surveys could also have been used. The reason for not using them was the time constraint and the constraint that people needed to be scheduled for interviews, which inevitably interfered with their work.

The interface was designed using rapid prototyping. It was evaluated in iterations with a period of redesign after each evaluation period. This approach was chosen because it is a user centered approach that is favored by Benyon et al[11]. It is also the approach used in agile software development methodologies[16].

The tools chosen for the rapid prototyping was Flex Builder 3 for making the Flex interface. Microsoft Visual Studio 2008 and C# was used, as well as Microsoft SQL server, for the backend web service and database logic.
Common ASP web service was used instead of Windows Component Framework (WCF) since .NET version higher than 2 was not supported at the web server at the time. It was also unclear whether Flex supported WCF services or not.

Before deciding on Flex as the front end application framework, other similar frameworks were considered as well. The considered frameworks were OpenLazlo, Flex and Silverlight (latest beta version).
A basic prototype using Silverlight was also made during the requirements gathering stage, but had to be discarded since it required additional installation of software on the clients. OpenLazlo was a good candidate, but was decided against since Flex builder from Adobe had a better IDE. My opinion is that Silverlight had the best IDE, which is Microsoft Visual Studio 2008. This is of course just my own personal opinion.

A complementary approach to the user interface design I used could have been to base the interface design not only on PACT analysis and requirements gathering and general good design principles, but also on established user interface design patterns. Design patterns have concrete solutions to concrete problems[17]. In “Designing Interfaces: Patterns for Effective Interaction Design”, Tidwell explains patterns applicable not only for web and desktop applications, but also for flash applications for example[17]. I was unfamiliar with interface design patterns at the time when the application started to be built and so did not use them.

2.5 Interface evaluation

Before each user evaluation period, the interface underwent an “expert evaluation” by me, the designer. That is not an ideal way to do expert evaluation, as tests has shown that only about 35% of the design flaws can be discovered that way[11]. This will be further discussed in the discussion chapter.

The interface evaluations were conducted with one employee at a time that used the interface of the manual. The test person was asked to be verbal about what he did with the interface. A sound recorder was used to capture the details of the users experience with the interface.
The evaluation was supervised by me, and I gave some instructions to the user in what they had to do and offered help when the user got stuck, as is recommended[11]. The user was also free to use the application as he saw fit, in an unstructured manner.
One other possible method could have been pair evaluation where two users together help each other when they encounter interface problems. The method was discarded however, due to resource constraints.

The results of the evaluations were then acted upon by redesigning the interface.

3. The interactive design

User interface design decisions are explained.

3.1. Target users

The users that were supposed to use the TBI manual were employees and consultants at Sandvik that quickly needed to get a grasp of what TBI was. To be more specific, these were the possible user types for the application:

- Employees and consultants at SSD doing development work on integration components.
- Employees and consultants at SIT and SSD working with system integration implementations.
- Customers to SIT and SSD doing preliminary studies of integration solutions. The customers are the other companies within the Sandvik group.

The user groups were quite uniform in nature and were skilled computer users.

3.2. PACT analysis and requirements

As is said in “Designing interactive systems”[11], the use of stories is a good method to capture requirements. Stories were however not used.
Instead a PACT analysis was made with the focus of achieving the goal without spending too much time on requirements gathering. Instead the focus was on gathering requirements as the prototype evolved, since people might not know what they actually need before there is something more real to talk about. That is one reason why interactive design is iterative[11].

The people part of the analysis is already covered in the previous chapter.
The analysis was based on observation of the intended users.
Requirements also came from other sources as stated in the method chapter above.

The activities that the interface should support were:

- Allow for users to be able to quickly navigate to any portion of the manual.
- Allow any content to be updated by a user.
- Allow for the inclusion of multiple types of content, like videos, pictures, text and diagrams.
- Allow diagrams to be edited by the users.

The work environment on the computer for every user was a very homogeneous one since Sandvik has a standard user client based on windows XP.
The context for the application was that the user should be using it when sitting in front of his computer when doing his usual work. He should be able to quickly access certain information as needed, or be able to start the application and take time to learn about TBI in general.

The technology had to support these requirements:

- The manual should not require installation of any sort.
- The manual should be available from any networked PC.
- There should be some form of central storage of the content so that the users could update the content for others to see.
- Video, picture and text should be able to be handled.
- The diagrams should be dynamic so that the user could change them as needed.

3.3. The design idea

In order to satisfy the initial requirements a creative leap was made. It is a necessary step many times in interactive design[11]. The idea was to use dynamically created diagrams as a way to easily express information in a graphical way without requiring the user to use external software.

The idea was that information then could be presented both in a visual form in a diagram and then further explained in picture, text or video content, accessed from each node in the diagram. The users should then be able to easily add information to the application. The reason why multiple kinds of media was desired, as discussed earlier, is that the human memory is multimodal and stores information in many forms[11].
Worked examples could in some circumstances be a more effective way to learn something than to let the users themselves try to solve a problem in order to learn something[18]. Worked examples may be presented not only in text format but also as a visual demonstration[19], which is a reason why it might be beneficial to incorporate instructional worked example videos in the manual.
More than one kind of dynamic diagram was desired, as some diagrams fit certain information more than others.

3.4. Iteration 1

The first version of the interface was static in the sense that the user could not change any content[20]. It showed a diagram and some text in a web browser. The diagram was clickable and when clicked presented the user with a new page of text[21].
Users understood how to use the interface, but there was a need to make it dynamic, as also stated in the requirements earlier. There were two users testing the application.
The evaluations seemed to confirm that diagrams used for navigation was a feasible design idea.

3.5. Iteration 2

The interface was redesigned in order to be able to use dynamic diagrams. The solution took the step from static html to a flex application with a backend server running web services and a database. The dynamic diagram was based on code from the flextuts library[22] and was a hierarchical diagram. The testing was done using only one test person[21] due to time constraints.
The user evaluation showed serious flaws in the application. The user needed help to understand how to use the application. The interface used buttons that changed the current state of the application.
The user could for example press the edit button. Every click was then interpreted as an edit action. Also, clicks outside of the nodes generated an action. So a click acted on a node even though it was not under the mouse pointer.
That apparently went against the users mental model of how such a web based application should behave.
Also, the user expressed a need to have some form of menu for navigation. One other issue was that the user wanted to get an idea of the content behind a node in the diagram before he clicked it.

3.6. Iteration 3

The flex application was redesigned so that it no longer used states. The user was always in a navigation state. To do any edit, add or change action, the user used right click to access a context menu for the diagram node that at the moment was under the mouse pointer. It was also not possible to click anywhere but on a node within the diagram. The user testing was done with one user[21].
The application was instructional in that it prompted the user with information about the existence of the right click menu, when hovering above a node in the diagram.
Also the diagram used text icons that in English described the content to the user.

The design appeared to more closely follow the mental model of the user. There were however issues. The icons were bitmapped text in a certain italic font. It was not perceived as a good solution. Instead it was suggested to use a graphic icon to deliver the information about what kind of content that was behind the node.

The user also needed a way to edit content and still have some preservation of content or undo functionality, so that a user not by mistake could erase a lot of content.
It was suggested that a supervisor should be notified on erases, and the actual erasing should not take place until a privileged user authorized the action.

3.7. Final design

A crumb trail system was implemented in order to make it possible to get a sense of where in the hierarchy the user was. It was an effort to remedy the need for a menu.
Other styles for the icons were used so that there was no more bitmapped text but small pictures used instead. The time for the thesis ran out, so all of the needed changes were not implemented in the final design.

4. Results

The three main components that constitute the result are presented.

4.1. Manual

The text and picture content for the manual were written as a common printable document, and is included in the appendix. The content was also inserted into a running instance of the interface to the manual.

4.2. Interface

The goal for the interface was to help the user learn the information viewed through it. It should have functionality that was easy to use, and help users that wanted to edit the content to express information in diagram and text form, as well as static pictures and movies.
The result did not fully reach the goals.

These general goals of the design were met:

- There was rapid feedback given on all actions involving possible long waiting times when communicating with the server.
- The visibility of content was provided for both with scrolling and zoom functionality.
- Since the interface relied on existing Flex interface designs for buttons and other components, it was clear what actions they afforded. Also, the dynamic diagram nodes afforded being clicked on.
- The user was presented with an interface that did not require the user to remember information. At least not more than 3-4 facts, as is the recommended limit for a user to have to remember[11].
- The dynamic diagram was a success in the sense that the user found it easy to add graphic content as a diagram.

These were the problems of the interface application:

- The navigation of the interface did not completely follow the user's mental models of the application, which is to say it had a navigation method that was unorthodox.
- The interface had no built in recovery functionality. There were also no constraints when deleting content.
- It was in no way designed with disabled people in mind.
- The Icons was not familiar to the user, as they were custom made, and the interface in general was not familiar to the user.
- There was no flexibility in how a task could be accomplished.
- Search engines could not index content. This is however not such a sincere problem as it might appear as the information is stored in a database and may be displayed to search engines in some fashion that makes it possible to index it, like having a html frontend for search engines. It was also not a design requirement.

4.3. Backend

The backend asp web service worked well in a single user environment.
However, the backend application had some flaws that prevented it from being used in a larger scale. It lacked versioning functionality. It lacked transactional functionality to make sure the application worked in a multi user environment. It had a way of lending out identification tokens for new content that resulted in a database that slowly increased in size. That could have been remedied either with some cron job that periodically kept the database in shape, or a redesign of the whole application.

5. Discussion

The design and the results are discussed.

5.1. Why not a perfect result?

When designing interactive systems it is very important to have a user centered approach[11]. That is why the PACT analysis was conducted. The design process however, had a lack of involved users. Since there were not enough people involved in the solicitation of requirements and in evaluation of the design, it had to somehow show in the final application, as it did.

As is recommended in "Designing interactive systems"[11], a combination of expert and user evaluations should be used. One expert evaluator will only be able to find about 35% of the faults of the design. Nielsen suggests[23] that the optimal number of evaluators is around 3 to 5.

The problem was also related to the fact that I found no specific design guidelines for making flex applications. Much has been written on design guidelines for web pages and desktop applications. But with web2.0 applications, the fine dividing line between web and desktop user interface design is blurred. They have the properties of both to some degree, since they run on the client computer and might have access to local storage on the computer, but are presented in the web browser window.
There is of course general design guidelines such as those described in "Designing interactive systems"[11], and that was the guidelines I followed. There is literature that discusses design guidelines for flash applications[17], which is the final result after compiling a Flex solution, but at the time of design I was not aware of them.

One interface design pattern in particular that should have been beneficial to use was the visual framework design pattern[17]. It states that the look and feel of the interface should be the same in the whole application, but afford different content to be displayed in it[17].

5.2. Design decisions

The interface was supposed to support learning. One design objective was then to present the user with the information in different kinds of formats, since humans remembers easier when the information is coded in more than one way, like when the information is coded in both text and picture instead of just text[24]. For example, to both see a diagram expressing some information and also present the same information in text. Multimedia in itself is however not a magical solution for more efficient learning, as Mayer says[24].

The interface should never present the user with a wall of text, as that scares the user[25]. Instead the interface should use colours and the text should be divided into paragraphs since that makes it easier for the user to absorb the information. The interface application for the manual made it easy to avoid by having a rich text editor that allowed for a colorful and pleasant presentation of text.

A good design guideline is that the interface should be instructional rather than personal, since users tend to get irritated on personal interfaces[11]. That led me to incorporate several instructional messages within the interface in order to guide the user.

The conviviality is important to a user[11]. A friendly interface is what a user may want to use for a long time. The tone of the messages in the interface is designed with conviviality in mind.

A uniform way of handling input is a good way to handle input[11]. People should not have to remember how to interact with the application at particular places. A good interface should only require the user to remember around 4 facts[11]. The final design of the interface for the manual had a quite uniform way of handling user input, but it was not perfect. It was revealed during testing that the user expected to be able to access edit functionality by right clicking, even though there was a button used for that purpose instead. That was contrary to how the rest of the application worked.

5.6. Related works

There are numerous works related to developing computer interfaces, but there are fewer that talks about system integration in combination with human computer interaction.
One such work is the thesis “Systemintegration på vägverket”[26], where the author talks about HCI problems when deciding upon an enterprise resource system, in a system integration context.
There is works related to e-learning and how to use HCI and information and communication technology to assist learning. One such work is “Computer assisted second language learning”[27].
One paper has shown in an empirical study[28] that the use of interactive technology used in knowledge transfer might be beneficial for enhanced learning in some cases. The article compares learning with only a text based presentation, a text and visual presentation, and a presentation where the user could interact with an application while learning[28].

6. Appendix

The document version of the manual is presented.

6.1 TBI manual

TBI Manual

Date: 18 June 2008

Author: Jonas Jakobsson jonas@joyofbits.se
Made as part of the "Integration for dummies" thesis project, which may be viewed at www.joyofbits.se/thesis2008

The content of this document is the TBI manual, which also consists of the presentation interface.

Thanks to Patric Ferry, Kent Eklund, Anders Rackner, Thomas Lundström, Magnus Wiksell, Niclas Gabrielli and everyone else at ICC for all the help.

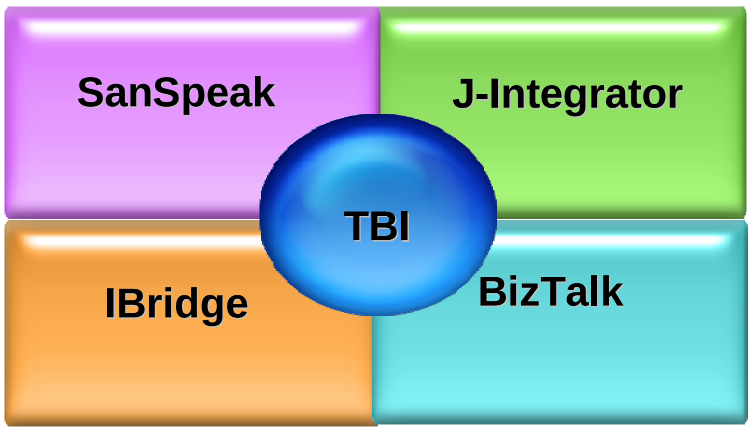


Table of contents

1. Overview

1.1. Why use system integration

These are examples of when there is a need for a system integration solution:

- A new company is acquired by Sandvik AB and a host of new computer systems needs to be integrated with existing computing infrastructure.
- There is a need to access information in some part of the organization from a new place or access it in a new way or modernize an existing integration solution.

System integration is quite simply the process of integrating systems so that they may connect and communicate with each other. The methods to do this have evolved over the years. Nowadays you probably hear abbreviations like SOA and ESB when you talk about system integration. Point-to-point integration solutions are very rare these days. Instead you typically use some integration concept like TBI that will make integration work as easy and efficient as possible.

Sandvik Systems Development (SSD) has a concept of how system integration can be achieved that is called TBI, which stands for Total Business Integration.

1.2. Overview of system integration at Sandvik

Total Business Integration (TBI) at Sandvik is a concept that comprises a number of software systems. Each system has a particular role in a system integration implementation.

To achieve business integration a combination of these systems are typically used.
In some cases development work is needed in order to achieve integration. In some cases there is a need to develop plugins to the existing integration components.

There is typically both SOA and ESB components in each integration solution. Common standards and standard transportation methods are used.

In typical use, the systems are just installed and then configured to handle the integration task, except for Jintegrator, which is not a ready made system but rather a structure of how to implement a part of a system integration solution in particular cases where java is used. Configuring a system integration system typically means specifying routing rules for messages and transformations to be done on the content of the messages.

Typically a number of such integration systems are used together to form a working integration solution. Each system works like a link in a chain connecting the systems that needs to communicate in the new integration solution.

Usually only a subset of the available integration systems are used in any integration implementation.

1.3. Workflow of a typical system integration solution

Typically the system integration is about transforming a message to xml if it is not already in that format. After that the message is sent over a transport to some destination. The destination might for example be a client computer system. On the way it may pass over a number of integration components, like iBridge, SanSpeak and BizTalk.
Message routing is set up at the components in the integration solution. Routing can be done in all the covered TBI components described in this manual, but routing in BizTalk is the preferred location to place complex routing and orchestration at.
A typical integration solution is made mainly through configuration.

1.4. Common transports

There are many possible transport protocols and transport systems in use in TBI.
Some more commonly used ones are:

HTTP/HTTPS – Used in web integration and web service integrations.

MQ – Websphere MQ is used, not Microsoft's MSMQ.

File – This might be a flat file or a XML file.

FTP – A lot like the File transport but transported to some remote server using FTP.

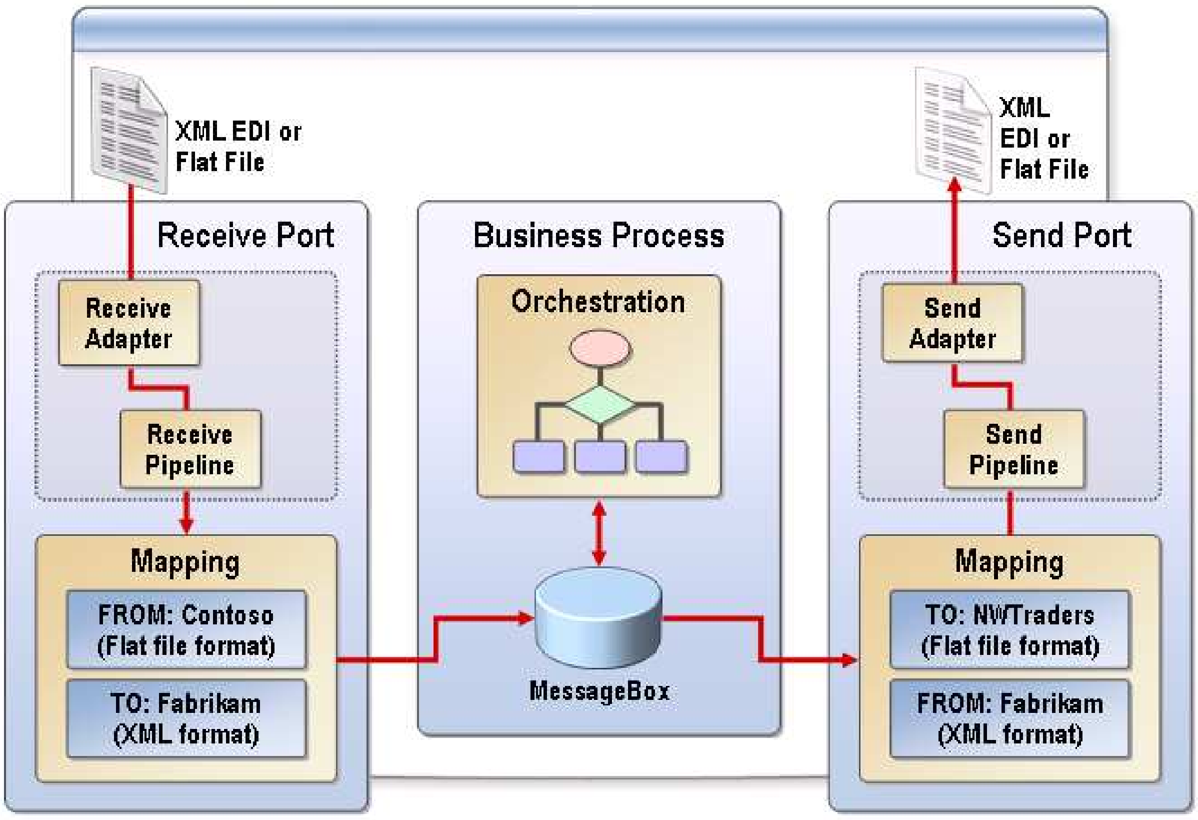
2. BizTalk

2.1 Overview

BizTalk is, together with iBridge, the central component in the TBI concept.
Its main use is message transformation and routing of messages.
Each integration solution with BizTalk is made in visual studio. You may either use preconfigured functions on the message in order to transform it, or apply a self made xslt.
When writing xslt, one might want to use an external editor, like XMLSpy, instead of the built in editor in visual studio, since the built in xml editor (visual studio 2005) is very limited.
Besides transformation, it is also possible to do very complex routing of messages in BizTalk.

With orchestration in BizTalk it is possible to make several involved steps that are not possible with any other component in TBI.

This picture gives a good overview of the functionality of BizTalk:



(Picture taken from introductory ppt-file for BizTalk, 2933A_01.ppt)

The business rules engine in BizTalk is not used.

Messages that come into BizTalk are first processed by an input adapter. Then the message is put on the message queue. Output adapters that has subscribed to certain types of messages will retrieve the message from the message queue and perform operations on it.
Several output adapters might have subscribed to certain messages, and so the message might be processed by multiple output adapters.

BizTalk together with iBridge forms the ESB (enterprise service bus) of the TBI concept.
An ESB is the center of a star-like formation of systems. Each system connects to the ESB.
The reason for that is that each system only has to be integrated with the ESB. After that, any system can communicate with any other system connected to the ESB.

2.3 Platform and dependencies

BizTalk runs on some Windows version above or equal to 2000. Typically Windows Server 2003 at the moment.

2.4. When to use it

BizTalk is used in these scenarios:

- When the messages may have to be transformed, like when a date is needed to be changed from some format to another.
- BizTalk might be an alternative if routing of messages is required.
- When complex routing, orchestration, is needed.

2.5. How to use it

BizTalk only communicates with iBridge on both the input and the output adapter. In BizTalk the input and output adapters for iBridge are called “submit” in the visual studio IDE (integrated development environment).
BizTalk may send messages to MOM (Microsoft Operations Manager), like log and error messages. That functionality is also provided by iBridge and is configurable.
Each integration solution is made in Microsoft visual studio.
After a solution is made, it is administrated from the BizTalk administration console.

2.6. Adapters

Since the TBI convention says that BizTalk only communicates with an iBridge instance, it is not necessary to use any other input or output plugin than “submit”.

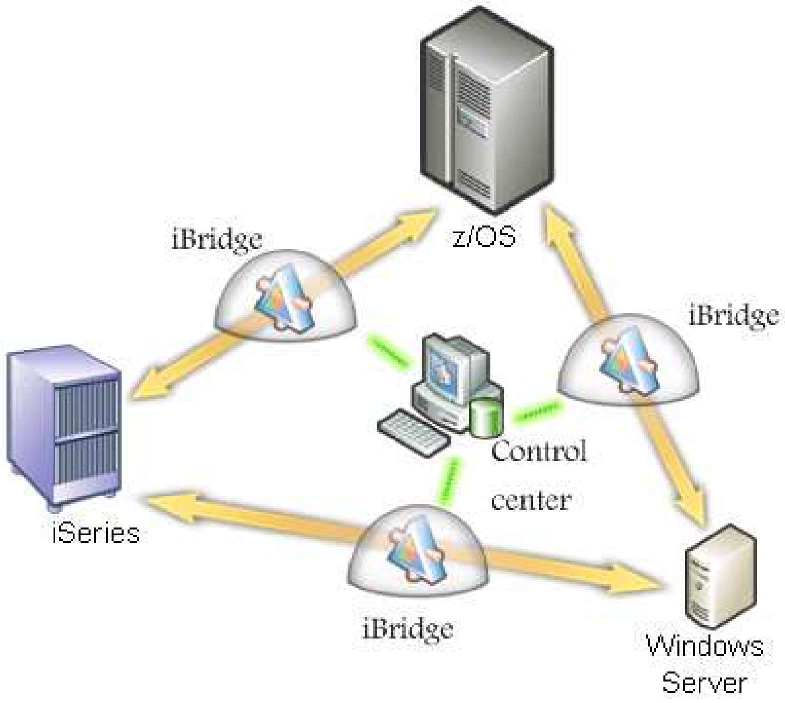
3. iBridge

3.1. Overview

iBridge is a system developed at SSD.
The life of iBridge started after an upgrade of BizTalk. The existing input and output adapters for BizTalk had to be rewritten. Instead of rewriting all the adapters for the new BizTalk version, a special program for BizTalk was made that took care of all the handling of different message formats.

That new program was later called iBridge. It also took care of logging and simple routing. The iBridge solutions are administrated through the TBI control center. TBI Control Center is a plugin driven application where iBridge is one plugin. Others are SanSpeak and Mobility Through that interface the whole integration solution may be monitored in real time.
The TBI control center may manage several iBridge instances in a distributed fashion.
The configuration for each iBridge instance is managed from the TBI control center.

This is a picture that shows the distributed way in which the TBI control center operates:



(Picture taken from power point presentation CustomerThemeMeeting_TBI_060208.ppt)

There are three ways in which to use iBridge:

1. Use existing input and output adapters. Everything is configured. No programming.
2. Write your own input and/or output adapter.
3. Use the iBridge API to access it from Java or .Net.

A message that iBridge should handle first arrives at the input adapter. The message then passes a preprocessor plugin, if configured to do so. After that the message is statically routed and processed by an output adapter.
If there is some form of recipe or reply of the message that needs to be communicated back to the caller, then that reply is processed by the output adapter, the post processor and then the input adapter, in that order.

iBridge may log to a central log. It may also send notifications through MOM.
It will automatically forward problems to the appropriate parties, as configured.

3.2. Platform and dependencies

A windows version equal to or newer than Windows XP or Windows Server 2003.
The .NET framework version 2 is required.

3.3. When to use it

iBridge is always used when BizTalk is used, and iBridge might be used standalone.
It should always be considered when a new integration solution is needed.
It is used when message translation from one format to another is needed.
It is primarily used with static routing. However there is support for simple dynamic routing.
It also provides central logging and configuration.

3.4. How to use it

In order to make a new integration solution with iBridge, a new flow is created in TBI control center.

When setting up the new flow, among others, these parameters are specified:

- Who should be notified of failures in the integration?
- What in and out adapters are to be used.
- What pre and post processors to be used.
- What the static routing should be.

The TBI control center then takes care of deploying the iBridge instance to the target server.
The iBridge instance uses a local xml configuration file that is created by the TBI control center.

3.5 Adapters

The adapters, or plugins, form the basic important functionality of iBridge.
It is through them that iBridge achieves its application to application integration.
The input adapter, also called the source plugin, receives the message.
The output adapter, also called the destination plugin, sends the message to some destination.

Between the input adapter and output adapter there is routing and also the processing by what is called the pre and posts processor.

There are a number of input and output adapters.
There is more output adapters than there is input adapters.

The input adapters are:	The output adapters are:
• MQSeries • SQL Server • File • .Net • Java • Log • Idoc (SAP) • SaPPCom	• MQSeries • SQL Server • File • .Net • Java • Log • Biztalk • Domino • Idoc • SaPPCom • OTMA • SAP Bapi x 7 • SMTP

4. SanSpeak

4.1 Overview

Sanspeak was developed at SSD.
Its main purpose is to work as a gateway, placed at the border of an internal network (DMZ or demilitarized zone).
It receives web service calls and delivers the message to the internal network. When delivering the message to some destination on the internal network, the message may be transported over one of a number of different transports.
When sending a web service request to SanSpeak an authentication is made to a certain user. That user may be part of some group.
The message is routed based on user, group or content of message.

SanSpeak has a database where it stores logs. That database may be centrally monitored.

4.2. Platform and dependencies

A windows version newer or equal to windows 2000.
The NET framework version 1.1 is used. SanSpeak is as of now not ported to version 2.

4.3. When to use it

SanSpeak is used when there is an internal network where there are resources that may not be exposed to the outside directly, but where there are external systems that needs those internal resources.
It may also be used as a web service communication bridge.

4.4 How to use it.

No compilation is typically needed. To make a solution, the configuration is written to an mssql database. Then an instance of SanSpeak is run on an appropriate machine, indicating where the configuration database is.
Each SanSpeak solution is maintained through the TBI Control Center.
In that program, things such as routing rules are specified.
Notable is that if one wants to do message based routing, then the group based routing table in the configuration dialog is used.

5. Jintegrator

5.1 Overview

Jintegrator is not a ready to be used as an off the shelf integration system, but a set of recipes on how to create java based integration solutions. Those solutions are needed primarily at mainframe systems.
When doing an integration solution using Jintegrator, you compile more than configure, as opposed to iBridge and SanSpeak.
You will also need the castor code generator, Maven and other software related to Java compilation.
Maven is used as the building tool for any new integration solution with Jintegrator, but there are existing solutions that use Ant as the build tool.
Each solution tends to be unique, but there are some common attributes:

- Each solution is using XML. Anything going in or coming out from the mainframe is made into XML by Jintegrator before being passed on to any external systems.
- Logs are centrally stored in a database.
- The XML code for the java solution is created by the help of the castor code generator.
- About 80% of the total amount of code for the Jintegrator solution is automatically generated.
- There is a high level of code reuse from one implementation to another.
- The solution supports communication over web service, http and MQ.
- A Jintegrator daemon is the typical result of a Jintegrator solution, and runs on the mainframe.

- There is a XML configuration file that the daemon reads.

A very important part of any Jintegrator solution is the storage of the solutions code. It is typically stored in a source code repository. It used to be CVS, but is at the moment a Subversion repository.
When a change is needed to be made to an existing solution, a copy of the source tree for the solution in question is taken from the repository. It is then modified and submitted to the repository again for future reference.

The Jintegrator solutions are typically not 100% cross platform compatible. For example, the name of a running process at the mainframe is used to pass information about where the current XML configuration file for the Jintegrator solution is located.

Jintegrator is typically centered on the daemon, as it is used as a way to avoid having to start a new instance of Jintegrator for every message that arrives and needs to be processed.
This is something that WAS (Websphere application server) will remedy as it will remedy the problem of high system load caused by frequently starting and stopping a Jintegrator instance.
WAS is about to be launched at Sandvik at the time of writing.

5.2. Platform and dependencies

The platform is AS400 and iSeries, having java installed.
In order to create a solution, the Java SDK is needed, together with a source code repository client like Subclipse or TortoiseSVN. Also the Castor is needed for code generation.

5.3. When to use it

Jintegrator is for example used on mainframes with programs that when they were written did not have much in the way of communication or any API that could be accessed remotely.
It is preferable to develop new solutions with Websphere application server (WAS).
It is needed when the application at the host server does not have web service or MQ communication capability.

5.4. How to use it

To develop a new solution one typically takes an existing solution and modify.
Existing solutions are found in the Jintegrator source code repository.
Each new solution and each new version of each solution should be stored in the source code repository.

Typically you modify the XML configuration file for the daemon. In that configuration you may specify things such as the routing rules for the messages.

7. References

[1] “Information wants to be free” http://en.wikipedia.org/wiki/Information_wants_to_be_free visited on 5 August 2008
[2] “Parkinson's Law: The Pursuit of Progress”, C. Northcote Parkinson, London, John Murray (1958)
[3] “Parkinson's law” http://en.wikipedia.org/wiki/Parkinsons_law visited on 5 August 2008
[4] "Enterprise application integration", Gable, Julie, Information Management Journal (March/April 2002).
[5] The Sarbanes-Oxley Act of 2002 (Pub.L. 107-204, 116 United States Statutes at Large. 745, enacted 2002-07-30)
[6] Report on the first quarter 2008, Sandvik AB, http://www3.sandvik.com/pdf/ir/Q1-08_web.pdf visited on 5 August 2008
[7] A speech given at a course for new employees at Sandvik, Management Director Sandvik Information Technology, Håkan Sundin, (May 2008)
[8] Press release 7 september 2007, Microsoft, http://www.microsoft.com/sverige/pr/articles/2007/0709073.mspx visited on 5 August 2008
[9] “The homepage of this thesis” http://www.joyofbits.se/thesis2008 visited on 5 August 2008
[10] “Academic Archive On-line” http://www.diva-portal.org visited on 5 August 2008
[11] D. Benyon, P. Turner, S. Turner. *Designing interactive systems*. Addison-Wesley. (2005)
[12] Interview with Magnus Wiksell, Sandvik Systems Development. (2008)
[13] Interview with Thomas Lundström, Sandvik Systems Development. (2008)
[14] Interview with Niclas Gabrielli, Sandvik Systems Development. (2008)
[15] “Forskningsmetodik – om kvalitativa och kvantitativa metoder”, Holme, Solvang, Studentlitteratur, Lund (1997)
[16] “Principles behind the Agile Manifesto”, http://agilemanifesto.org/principles.html visited on 5 August 2008
[17] “Designing Interfaces: Patterns for Effective Interaction Design”, Jenifer Tidwell, O'Reilly Media, Inc. (November 21, 2005)
[18] Sweller, J. (2006). The worked example effect and human cognition. Learning and Instruction, 16(2) 165-169
[19] Lewis, D. (2005). Demobank: a method of presenting just-in-time online learning in the Proceedings of the Association for Educational Communications and Technology (AECT) Annual International Convention (vol 2, p. 371-375) October 2005, Orlando, FL. http://www.coedu.usf.edu/agents/aect2005/dlewis_aect2005paper.pdf visited on 5 August 2008
[20] Interview and application evaluation with Anders Rackner, Sandvik Systems Development. (2008)
[21] General TBI related interviews and application evaluations with Patric Ferry, Sandvik Systems Development. (2008)
[22] The flextuts library, http://www.flextuts.com/ visited on 5 August 2008
[23] ”Usability engineering”, J. Nielsen, Academic press, New York (1993)
[24] Mayer, R. E. (2001). Multimedia Learning. New York: Cambridge University Press.
[25] ” Top Ten Web-Design Mistakes of 2002”, http://www.useit.com/alertbox/20021223.html visited on 5 August 2008
[26] ”Systemintegration på vägverket”, 20HP credits, thesis work, Petter Midtsian (2007)
[27] “Computer assisted second language learning”, Katerina Kalimikeraki, Iordanis Kavathatzopoulos. I volume 1, number 2 of Open and Distance Education and Education Technology, Propobos Publishing, Athens, pp 74-97, 2005.
[28] “Learning Newtonian mechanics with an animation game: The role of presentation format on mental model acquisition”. Chan, M. S., & Black, J. B. (2006). Paper presented at the American Education Research Association Annual. San Francisco, CA.