Reflection of a Year Long Model-Driven Business and UI Modeling Development Project

Noi Sukaviriya[1] , Senthil Mani[2] and Vibha Sinha[2]

[1] IBM TJ Watson Research Center
[2] IBM India Research Lab
noi@us.ibm.com, {sentmani, vibha.sinha}@in.ibm.com

Abstract. Model-driven software development enables users to specify an application at a high level – a level that better matches problem domain. It also promises the users with better analysis and automation. Our work embarks on two collaborating domains – business process and human interactions – to build an application. Business modeling expresses business operations and flows then creates business flow implementation. Human interaction modeling expresses a UI design, its relationship with business data, logic, and flow, and can generate working UI. This double modeling approach automates the production of a working system with UI and business logic connected. This paper discusses the human aspects of this modeling approach after a year long of building a procurement outsourcing contract application using the approach – the result of which was deployed in December 2008. The paper discusses in multiple areas the happy endings and some heartache. We end with insights on how a model-driven approach could do better for humans in the process.

Keywords: Business process model, model-driven user interface, UI design, solution design

1 Introduction

Model-driven techniques have been applied in many areas such as software engineering, software architecture, service-oriented architecture, user interface development environments, and recently business process modeling and design. While model-driven techniques enable information organization and downstream automation, their benefits to human are enablement of automated analysis and the expressive power at the level that is closer to the users' conceptual models. In our work, we bring about model-driven benefits in two domains – business process modeling and user interface design.

Model-Driven Business Transformation (MDBT) is the work that our collaborator has been carrying out for a few years prior [1,2] with several successful completed and on-going customer engagements [3,4,5]. MDBT supports business process modeling using an artifact-centric approach [6] – the approach which will be briefly described in the next section. Once a process model is defined, it is transformed into an executable IT solution. While the MDBT framework allows business owners to dictate and essentially specify their IT solutions, it also reduces the time to develop the basic application logic such as business flow and the retrieval and storing of business data. With this approach, what remains time consuming is building solution UIs to go with the business logic and flow that are automatically created.

In 2006, we created a vision that a model-driven UI development environment would reduce the UI production time in the context of business design. Our first goal was to develop a methodology that guides UI solution design to meet business design goals, and continuing to meet changing goals through iterations. Our UI specific goal was to automate the tedious part of the design process while still enabling the UI designer to freely express their design. We want to pass the design integrity from UI designers directly to the solution developers. We want to support design simulation that can be demonstrated live to business stakeholders without implementing the backend business logic. We also want to build a platform which lends itself to reusable assets. While these goals are not fully realized currently, the work and the experience reported in this paper reflects a step towards these goals.

By September 2007, our model-driven UI environment [7] was ready to be tested. Our team at IBM research partnered with an IBM IT consulting team to apply the technology to build a real world application for a business customer. Our task was to build a global procurement application that enabled the process of drafting outsourcing requests, circulating for procurement inspections and approvals, facilitating supplier bidding, supporting multi-tiered reviews, and winner selection, and submitting purchase requisitions. The IBM IT consulting division was to be the recipient of the research technology transfer and would own the solution for the customer after the research team left. We released the pilot solution in December 2008 and the major release is on ramp for May 2009. IBM Research will remain involved in 2009 to assist the consulting team in delivering the second release that includes Asia Pacific countries. This second release would allow us to observe how the current model and tools adapt to additional complex business logic, which will definitely require challenging modifications to the current solution.

2 The Dual Model Approach and Methodology

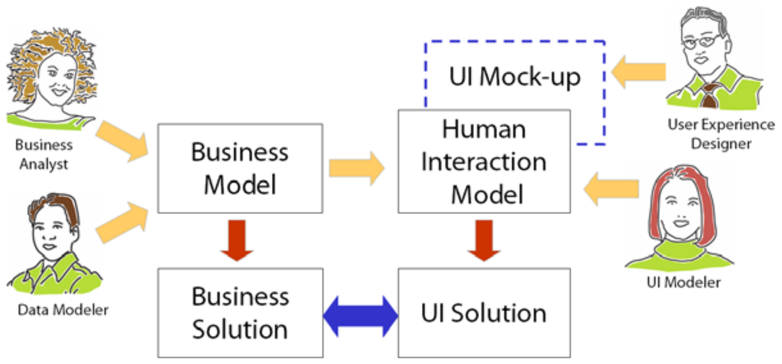


Fig. 1: Participants in the creation of a business model and a human interaction model

In our approach, business requirements were gathered at the beginning through business process modeling. Once requirements became sufficiently clear, a storyboard and UI mock-ups were created to confirm and further drill down on the requirements. The implementation phase differs in our approach in that the user interface is explicitly modeled (described in the following sections.) By following the methodology, executable business solution is created as business services. The UI solution is generated as a set of pages with appropriate calls to the generated business services. This work was previously presented in detail in [7.] Figure 1 illustrates key participants in the creation of the business and UI models. The total process involved a larger set of people including business stakeholders, subject matter experts, IT architects, project manager, and developers.

2.1 Business Modeling

In our approach, a business engagement starts with business process modeling workshops in which business analysts, subject matter experts, and user experience designers conduct face-to-face meetings to identify key business artifacts and their life cycles. These workshops are essentially requirement gathering workshops. To give the readers a sense of the complexity of the procurement application in this paper, it took about 5 workshops, 3-4 days each, to get a good grasp of the overall business process. The workshops were led by business analysts while the research team helped guide them through the modeling exercise.

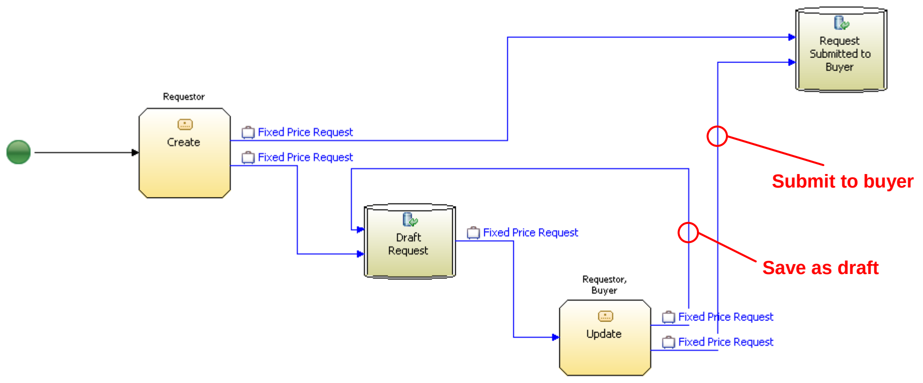
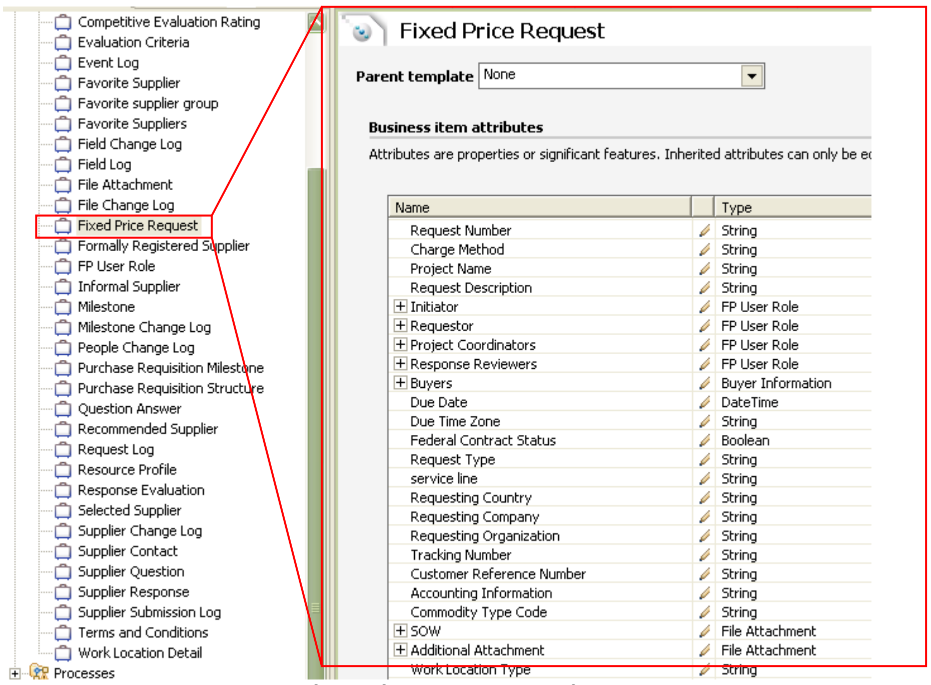


Fig. 2: An example of an artifact-centric business process model. The output ports of each task are labeled but not visible on the diagram.

The business process model was defined using a methodology called Artifact-centric modeling [3], of which focus is on modeling the life cycles of key business artifacts – artifacts that are the heart of a business. The methodology leads to a clean way of thinking about business operations without IT interference – by enforcing the thinking only around what happens to these key business artifacts. The use of artifacts as nouns and business tasks as the actions to the nouns naturally shakes the model free of IT-specific implementation such as button clicks or how data are navigated. The focus on key artifacts also helps avoid small tasks on non-key artifacts that may be specific to how a particular organization operates. The full life cycle of the “Fixed Price Request” for our application was drawn in several diagrams.

The application had two main artifacts - Fixed Price Request and Supplier Response. Figure 2 shows a snippet of the “Fixed Price Request” artifact life cycle. “Create” and “Update” are tasks and the users who perform the tasks are shown on top of the tasks. Though the artifact life cycle diagrams may appear visually similar to other process diagrams, the differentiations as mentioned is in the task semantics and granularity.


Competitive Evaluation Rating
Evaluation Criteria
Event Log
Favorite Supplier
Favorite supplier group
Favorite Suppliers
Field Change Log
Field Log
File Attachment
File Change Log
Fixed Price Request
Formally Registered Supplier
FP User Role
Informal Supplier
Milestone
Milestone Change Log
People Change Log
Purchase Requisition Milestone
Purchase Requisition Structure
Question Answer
Recommended Supplier
Request Log
Resource Profile
Response Evaluation
Selected Supplier
Supplier Change Log
Supplier Contact
Supplier Question
Supplier Response
Supplier Submission Log
Terms and Conditions
Work Location Detail
Processes

Fixed Price Request

Parent template: None

Business item attributes

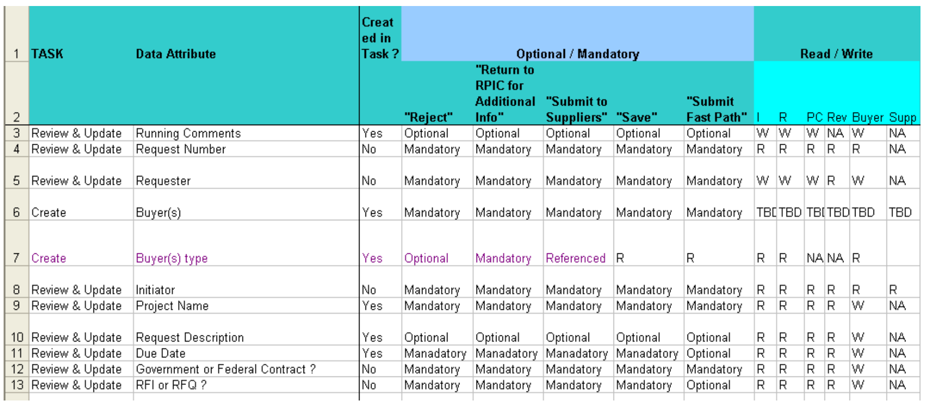
Attributes are properties or significant features. Inherited attributes can only be ec

Name	Type
Request Number	String
Charge Method	String
Project Name	String
Request Description	String
Initiator	FP User Role
Requestor	FP User Role
Project Coordinators	FP User Role
Response Reviewers	FP User Role
Buyers	Buyer Information
Due Date	DateTime
Due Time Zone	String
Federal Contract Status	Boolean
Request Type	String
service line	String
Requesting Country	String
Requesting Company	String
Requesting Organization	String
Tracking Number	String
Customer Reference Number	String
Accounting Information	String
Commodity Type Code	String
SOW	File Attachment
Additional Attachment	File Attachment
Work Location Type	String

Fig. 3. An example of the information model defined in the business model

As part of completing the business model, the data aspect of each artifact, as shown in Figure 3, is also filled in by business analysts and/or data modelers. This information is very essential for bridging the business model to the user interface model later on. We currently use IBM Websphere Business Modeler (WBM) as a front end tool to model business flow and the data model.

Our application has lots of requirements on when and whether certain user roles can create, view or edit the artifact data. The business team did not see how such requirements could be captured in the business diagramming tool. Complimentarily, a spreadsheet is used to capture these requirements. These readability and edit ability details of each business task from the business model were captured on an individual worksheet in the spreadsheet. Figure 4 shows an example of the read/write requirements for a task. Additional detailed descriptions of business tasks were kept in a Microsoft Word document, again, as the tool does not facilitate easy navigation back and forth among task descriptions and appeared to poorly support the conversation flow.



1	TASK	Data Attribute	Created in Task ?	Optional / Mandatory					Read / Write					
2				"Reject"	"Return to RPIC for Additional Info"	"Submit to Suppliers"	"Save"	"Submit Fast Path"	I	R	PC	Rev	Buyer	Supp
3	Review & Update	Running Comments	Yes	Optional	Optional	Optional	Optional	Optional	W	W	W	NA	W	NA
4	Review & Update	Request Number	No	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	R	R	R	R	R	NA
5	Review & Update	Requester	No	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	W	W	W	R	W	NA
6	Create	Buyer(s)	Yes	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	TBD	TBD	TBD	TBD	TBD	TBD
7	Create	Buyer(s) type	Yes	Optional	Mandatory	Referenced	R	R	R	R	NA	NA	R	
8	Review & Update	Initiator	No	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	R	R	R	R	R	R
9	Review & Update	Project Name	Yes	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	R	R	R	R	W	NA
10	Review & Update	Request Description	Yes	Optional	Optional	Optional	Optional	Optional	R	R	R	R	W	NA
11	Review & Update	Due Date	Yes	Manadatory	Manadatory	Manadatory	Manadatory	Optional	R	R	R	R	W	NA
12	Review & Update	Government or Federal Contract ?	No	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	R	R	R	R	W	NA
13	Review & Update	RFI or RFQ ?	No	Mandatory	Mandatory	Mandatory	Mandatory	Optional	R	R	R	R	W	NA

Fig. 4. A spreadsheet business users specified for the task "review and update"

2.2 User Interface Design

Once the requirements became sufficiently clear, a storyboard and the UI mock ups were created. There is nothing very new in this process, except the user experience designer in our team needed to be totally aware of the business process flow and the artifact data model.

A couple of points are worth mentioning here. First, our intention was to use the UI model to mock-up the UI and present it to the business team. However, the lack of a visual tool in our toolset prevented this exercise from happening in a timely fashion, hence this idea was dropped. Instead, we used a static PowerPoint storyboard to present the UI mock-ups. The storyboard provided synopses of what would be months of activities by multiple people (request drafting, the review process, supplier bid revisions, cancellations, exceptions, contract renewal, etc.) into hundreds of charts. Several multi-hour sessions were spent validating the UI with the subject matter experts. We found it still very hard to follow the storyboard for an application of this size with many variations. An interactive storyboard which allows use cases to be selected and followed would be more appropriate.

2.3 User Interface Modeling

After the subject matter experts validated the initial UI design, the UI modeling process began. We used our custom UI modeling tool to automatically extract user roles, tasks for each role, artifact details, and artifacts each role is entitled to access from the business model. The tool presented all of this information in the Human Interaction perspective. The technical detail of this tool was presented in detail in [7.] Let us briefly summarize here for the context of this paper.

The UI modeler had 3 tasks. The first task, which is optional, is to model the read/write access for each user role and for each specific task or group of tasks. This task is primarily encoding the spreadsheet inputs from the business team into a more optimized form in the tool. The tool allows added efficiency such as allowing specifications of similar access requirements across user roles and tasks. The specifications could have been done by business analysts but since the tool requires an installation of Rational Software Architect with a large footprint, the business analysts were more than happy to have this work done by the UI modeler.

The second task for the UI modeler was to model screens as pages associated with each user role. Each page was modeled as a hierarchy of page fragments, each of which contained either further fragments, or a layout and UI elements. While modeling each page, the UI modeler associated the UI elements to data attributes from the artifact data model. Alternatively, she can use the information model to automate filling in the content. For example, a set of data attributes can be associated to a UI container, and the system will automatically fill the container with appropriate UI elements. Had the modeler done the first task, the automatic generation can be more accurate in selecting appropriate UI elements for viewing or editing. These elements can be altered later if the modeler wishes. The connections from UI elements to data attributes are kept in the model hence allowed the system to generate appropriate service calls to fetch or store data at run-time.

Lastly, the UI modeler defined page flow as links between pages. Some links maybe coupled with a side effect of saving or removing data. Some may couple with outputs of business tasks (i.e. “Approve” output of a “Review” task) hence signifies business logic connections. The page content and flow, with connections to business data and logic, enables code generation to construct appropriate web service calls that fill the screen with business data and enact appropriate business logic.

2.4 Simulating the UI Design

With the UI model, UI code can be generated to render pages without the backend business logic. In our current implementation, the output from the UI model is an XML model feed to IBM Websphere Portlet Factory (an IBM product), which in turns generated JavaServer Page (JSP) code. When running in a simulation mode, our system pulled sample data from XML-based data sets. The data are selected based on the semantic types enhanced to the data model by the UI modeler [7.] This feature was designed to enable UI viewing without business logic and can be used to show low-fidelity UI to business stakeholders. It turned out that this feature was not sufficiently designed for its job. The sample data feed became confusing quickly after viewing a few pages since there was no continuity from one page to the next. Secondly, most of the use cases in our application were rather lengthy with many possible variations. The simulation could not present these variations in a way that made sense as it did not have application-specific data that would trigger the right behavior. We later implemented support for adding application-specific values to the sample data set. However, this was still insufficient as dynamic UI behaviors such as turning on/off portions of the screen or enabling/disabling buttons often depended on specific values. When these values were randomly picked, the dynamic behaviors became confusing. We need to enhance this simulation further to make it truly useful for business users. Positively though, the feature was heavily used by the UI modeler before we started backend integration as a way to see whether the basic layout and flow worked as planned. The UI modeler made a good use of this feature.

2.5 Finishing up the Solution

For the finale, we pulled together the generated outcomes from the business model and the UI model. Two products were generated from the business model – DB2 database definitions generated from the artifact data model, and business services based on the outputs of business tasks in the model. The database definitions were manually enhanced by the database administrator before they were used to generate the actual database tables. The business flow transformed into a business state machine and the business services were merely API for invoking state transitions. We will not dive into further technical details since it is not the interest of this community.

Additional development work was needed in this step to complete the solution. Business logic that was not based on flow need to be implemented, for example, routing a request to appropriate procurement teams, sending e-mail notifications to various parties when an event happened, calculations of some data values based on page inputs, etc. Also, since our UI modeling and generation is currently limited to static screens, dynamic behaviors needed to be programmed in JavaScript thus required additional UI developers. This is one area of our future work.

3 Related Work

Many business modeling languages exist such as UML, BPEL, CogNIAM, IDEF0, XPDL, [8,9,10,11,12] but one that is becoming a standard is BPMN [13]. While these languages provide variations of expressions and notations, with them come specific methodologies the languages enforce. The model-driven business modeling approach used in this work is rather unique in that the methodology coincides well with business user thinking, while the system can generate executable code. We do not intend to dive into related work in business modeling research here since we want to focus on the human interaction and human aspects of this work.

Many aspects of our UI modeling work have related work in the area of model-driven user interface environments. The use of task-driven modeling as a way to tie to UI model was reported in [14, 15, 16, 17, 18.] However, these research systems depend on the task and task hierarchy defined using the traditional HCI method [19.] The UI Pilot system [20] in particular was very close to our UI modeling environment in that it provided a 360-view of task, UI pages, users, and data components. Our work differed in that we extract such information from the business model and it remains so throughout the development cycle. In our methodology, UI modelers do not dictate the basis of the business tasks against which UI is to be designed. Some other related model-driven UI model and environments can be found in [21, 22, 23, 24.]

Our UI tool has an assistance that uses selected data attributes to fill in UI elements automatically. This work is similar to the work in the past in [25] though we use the semantic data types that are more familiar to business users (such as address, first name, last name, social security number, etc.) to drive UI element selection. We do not concern with a comprehensive automatic layout compared to [26.] Lastly, the specification of read/write access in the data model uses a similar concept as [27] with a conceptually similar hierarchy of read/write access. Again, our difference is the heavy tie to the business vocabulary and such a perspective it brings and enforces.

Lastly, new research work is emerging that ties business process modeling to UI modeling [28, 29.] We see our work as being rather similar though our work is guided by a business modeling methodology that prevents IT concerns or particular choices of interface design, hence the business model remains agile as UI design may change over time.

4 Analysis of the Methodology and the Process

In this section we discuss features that worked well and those needing improvements.

4.1 Business Model and UI Modeling Methodology Contains Solution Deviations

Business stakeholders found our approach extremely valuable. They resonated with the idea that the business model, which they helped create and refine throughout its evolution, provided a control over the solution. Business users can easily grasp the overview of the end-to-end process. The process remained at a level high enough for business users to identify where in the process that required modifications.

On one hand, the process model remained agile and all participants were comfortable with understanding the model implications. On the other hand, more detailed descriptions of what entailed the high-level tasks were captured deeper in the model. This was not seen by the business users. With business model as the original point of development and it remained so throughout the development process, and with the UI model integrated tightly with the business elements, many good behaviors followed. We observed that all team members talked in the same language – that was the language of the business elements. The artifact and attribute names given in the business model were used in the UI design language as well as by developers who programmed in cognizant of the same vocabulary. Business tasks were referenced all aboard. When a data attribute or a function was missing, it was easily traced back to its absence from the business model. Business analysts and UI design were then engaged to resolve the problems by changing the business and/or the UI models.

Interestingly, database administrators on the contrary requested the ability to map selected business data attribute names to other labels – specifically the table and column names in their master reference databases implemented prior to this application. The prompted reason was to reduce database personnel training since they were already trained to maintain these databases across applications. The database team members were aware of the mappings from the business vocabulary to their familiar database vocabulary, but this mapping had no effects on developers.

It was still possible to have communication breakdowns and wrong information entered in lower-level model. With a tight integration from the UI model components to the business model components, we had a control over the solution scope. All user roles were derived from the business model. Pages were assigned to appropriate user roles. The tool constrained each page flow to only link to tasks entitled to the user role owner of the page. On one hand, this did not leave room for misconducts in the solution behavior. On the other hand, a missing behavior could not be quickly added.

A real case was, for example, after a supplier declined terms and conditions of a bid, they could not reverse their decision hence were out of the bidding opportunity. The business team believed the implementation was simply missing. Upon inspecting the business model, it was clear that the intention was not recorded and probably never discussed. We noticed many cases such as this one occasionally in the process. This is one of those areas where, we believe, requirements for exceptions, as obvious as they may seem, often fall through the crack. The business model was a recorded agreement and a means to make it rather simple to find the missing requirements.

Our current approach is not free from errors such as communication breakdown. However, as seen over this project, the approach succeeded in limiting solution deviations in many directions as well as providing evidence of logical connections even if they were wrong. We continue to conduct research to better capture requirements, especially dynamic ones, in the context of artifact-centric modeling.

4.2 Iterations Must Work Well in Model-driven Environments

A model-driven environment means iterations are done in high-level models. Iterations must be well supported and must be designed in the tooling from the beginning. With our approach, we have observed iterations in operation all year long. It worked well in many areas and it ceased to work in some areas.

The UI modeling tool from the beginning had support for business model changes designed into the tool [30.] Since it must always be in synchronous with the business model, we made explicit decisions on how the UI tool would deal with different types of business model changes. We considered the tool working rather well. Throughout the development, the modelers and developers went through 2-3 iterations a day per UI model for a period of several months. The iteration was not as fast as we wished. Each version of the generated code needed to be recompiled and reassembled. With the industrial software, the type of server we deployed to, and the sheer size of the application, the time it took was unavoidable.

On the contrary, iterations on the business modeling did not fare as well towards the end. The intermediate UML solution model generated from the business model required additional enhancements before the code could be generated from it. The problem we had was two-fold. The tooling support for making such enhancements was not usable hence ignored by developers. Secondly, change management was not accounted for. This turned out to be a backlash later on in the development process. Every time the business model changed, the generated solution would wipe out the enhancements. Putting changes back would go from 1 day when the solution was still small to eventually more than 5 days as the implementation became larger. When the project schedule became tight at the end, we were forced to stop changing the business model, thus erasing the iteration benefits of a model-driven approach. Some key application problems cannot be resolved. The lesson learned here was that change management must be designed for and not treated as an accidental feature.

4.3 Business Users Cannot Express all that are Required

As mentioned earlier, our intention is to have business analysts and the business team owning the business process. Though our business modeling approach has been proven for a few years, business users in previous engagements were not active in handling the business process model with WBM. We attempted to achieve this goal early on in this project. Two observations we made in this endeavor. The ownership of the model slowly shifted towards the research team as the project got deeper in the implementation. This is understandable as making a business model deliver the correct specifications of business logic, and completing the information model details needed at run time is a rather daunting task. For example, in addition to basic data attributes given by business analysts, the information model was enhanced with unique keys, extraneous attributes and flags for system functioning. Implementing UI also required additional attributes for the overall usability that were not called for by business analysts. When the appearance of the data model started to vary significantly from the spreadsheets, the business team ceased their interest in the data model in WBM and only referred to the spreadsheets as the data reference.

One concept that was difficult for business users was data structure and modularity. Business users can understand data multiplicity (and some modularity) as in “there needs to be one or more milestones in the request before sending bids to suppliers” and optional/mandatory conditions as in “there can be multiple resource profiles but none is required.” Translating these requirements into 1..n or 0..n is not a big hurdle. However, structuring information into a module that can be reused was not natural to them. This is understandable. However, our approach mixes the need to optimize the database tables and the need for business users to enter and update the information model in the same tool. These two goals cannot be achieved by business users alone.

The business team also used less of flow model later on. When decisions to change model were made, they totally entrusted the research team to execute the changes and only occasionally checked the flow during discussions. We want to believe that they at this point were very familiar with the high-level model. Interestingly, the UI mock-ups became a more often used reference by the business team as we got deeper into the project. And that did not change even towards the end of the project.

4.4 Lack of a Visual Tool Changes the Nature of the Development Process

When we embarked on this research, we made a conscious decision not to build a visual editor. Our first priority was to prove the modeling concept and also we could not afford the time or programming efforts. This turned out to be a bad decision. First, the user experience designer did not want to and did not build the UI design directly in the tool; hence the design precision was not captured in the model as intended. We missed our goal in this project.

Lack of a visual tool had more ripple effects as it has proven to cause serious time consumption as well. UI modelers missed out on design precision as any developers would. Lack of a visual tool forced the modeler to work with text such as page names, UI element names, etc. Visually searching for a page was hard as we reached over a hundred pages. Page names were hard to remember even for the modelers themselves. Textual names also created personal workspace effects that made it hard for others to view the UI model.

Lastly, the IT consulting team viewed a tool without a visual editor as a programming tool; developers were assigned to own the UI model. 3 developers later on in the process took over the UI modeling as part of the technology transfer. We started seeing design deviations as in the traditional method. Most of all, we have experienced mismatches in the tool design concepts that were intended for other types of users.

4.4 UI Modelers Need to Understand Business Requirements and the Data

In our project, the user experience designer was involved with the design process from the beginning. However, the UI modeler was not. When UI pages were given to the UI modeler, the experience designer assumed it would be obvious to see the matching from the field labels to the attribute names in the data model. That was true perhaps about 80% of the time. Minor questions arose on data clarifications between the UI modeler and the experience designer. However, we had many pages that appeared similar using similar field labels but in fact needed data attributes from different sections of the model. We had many of those mistakes that were not detected earlier on but once detected, were already widespread and tedious to correct.

The connections from correct business data to correct pages had proven to be essential in creating UI pages that worked. However, using the data model with mixed initiatives (as mentioned in 4.3) can be quite technical and this task may no longer be appropriate to user experience designers. We need to further investigate whether the solution to this problem lies in a better data modeling tool that can support mixed purposes or in a better UI modeling tool.

4.6 Monolithic Model Prevents Parallel Development

When we designed the UI modeling tool, we did not think about multiple modelers in the same project. We were shortsighted. The outsourcing application reached over 200 pages across 4 user roles. At the beginning we decided to use two UI models to support two different style templates (internal procurement users vs. external users.) The separation did not suffice. As we transferred the UI modeling task to the consulting team, the task was assigned such that a modeler owned a particular user role. By storing the UI model in one big file, only one modeler could work on the model at a time. (Versioning tools could not resolve conflicts at the UI modeling level.) We temporarily separated the model further along the owner of the model but this solution was far from ideal. We learned that collaboration must be designed into the tool from the beginning, as it has many impacts on how the model is structured, the choice of modeling language platform, and the methodology of how team members would collaborate.

4.7 Who Should be UI Modelers?

When we started the project, we targeted user experience designers as the users of the UI modeling tool. However, lack of a visual tool took us off our intention. The year of experience had informed us however that our vision may be questionable. UI design skills are rare in IT consulting team and a project may go without such skills. What do we do? Connecting UI to the data model requires data modeling skills. Connection to business flow and the overall big picture requires business savvies. Designing quality UI requires visual and design sensibility. The lessons learned here is no one possesses such a range of skills. The modeling tool should look for opportunities to take advantage of the available skills while increasing automation to compensate for missing skills. Alternatively, a modeling tool should facilitate handing off from one user to the next, enabling multiple users with different skills to work together even in a single domain of UI modeling.

4.9 Generate What Matches Market Skills

Lastly, one of the complaints we received from the consulting team was the fact the UI code was generated as Websphere Portlet Factory (WPF) XML. This is not the skills that prevail in the market. Developers were not comfortable with the generated code which they did not understand. Moreover, when the generated code needed to be enhanced for debugging, developers were forced to learn the WPF programming environment. The consulting team needed to learn new skills to own the tooling. We have been requested to consider generating UI code such as JSPs and HTML to better match developer skills both as users and future developers of the tool.

5 Summary of Analysis

While we have no quantitative data, our IT consulting team had agreed that the project used much less developers and could deliver much more functionality within the same amount of time, in comparison with a traditional development method. We continued to allow business requirement changes very late in the development process, though not within the final months. We have seen positive attitude from business users who benefited from the ability to fine-tune the design after they had seen the running prototype. The design phase definitely had gone much longer in the development cycle. Had the business modeling iterations were possible through the end, business model changes would continue into the last months of delivery. This would not happen in the traditional development method.

Tooling design had great effects on the success of a model-driven project. We have seen concepts that worked well such as support for iterations, ease of technology transfer to practitioners with the UI tool, convergent to the same business vocabulary by team members, reduction in development efforts and time. We also had seen concepts that did not get sufficient considerations upfront that caused issues later, for example, sample data simulation, collaboration among modelers, and generated code that does not match the user skill sets. Throughout the year, our tool and the meta-model have gone through significant changes within the boundary that the project can afford. Major conceptual changes cannot be done during the project if we could not continue with the existing model. Lots of progress was made during the year to make the tool more usable both for the development process and for what it generates. One area, for example, which we did not discuss, was generating UI code that meets the accessibility requirements.

We have learned a great deal from the success and failure of our choices in the research tool design. Our model-driven development process is very promising. It provides users, with or without programming skills, the ability to look into the business model and/or the UI model to grasp the solution. It enables them to have a direct control over the solution. This has become a very positive experience from our engagement. A year of experience has changed the way we see the impact a model-driven approach have on practitioners. The dual modeling approach has impacts on a variety of practitioners ranging from business users, UI designers, to developers. Many of these issues were not in the scope of academic interests at the beginning. However, when one considers the absence of model-driven development tools in the marketplace, especially in the UI area, some of these lessons we learned we hope to inspire the readers to think differently on the impact of their work to the real world.

Acknowledgements: We would like to thank the extended teams who have been working with us for almost 2 years – the procurement business team Mark Kemp, Jim Carver, Cathy Ng-Pepe, Amelia Carruth, and Russell Parks; the IT consulting team Gautam Majumdar, Glenn Godoy, Marco Gomez, Sachin Popli, Manish Singh, and Raul Laprida. We would like to thank our own development team – Thejaswini Ramachandra, Ajoy Acharyya, Arindum Dutta, Sourav Chakrabarty, and Kalyani Deshpande for the untiring efforts; and the MDBT architect team Prabir Nandi, Terry Heath, and Florian Pinel. Finally, we would like to thank Anil Nigam for his consultant on artifact-centric modeling throughout the project.

References

1. Kumaran, S. Model-Driven Enterprise. In: Proceedings of the Global EAI (Enterprise Architecture Integration) Summit, pp. 166-180 (2004)
2. Kumaran, S. and Nandi, P.: Adaptive Business Objects: A New Component Model for Business Integration. In: Proceedings of ICEIS 2005: 7th International Conference on Enterprise Information Systems (2005)
3. Bhattacharya, K., Guttman, R., Lyman, K., Heath III, F.F., Kumaran, S., Nandi, P. Wu, F.Y., Athman, P., Freiberg, C., Johannse, L. and Staudt, A.: A Model-Driven Approach to Industrializing Discovery Processes in Pharmaceutical Research. In: IBM System Journal, Vol. 44 No. 1, pp. 145-162 (2005)
4. Bhattacharya, K., Caswell, N.S., Kumaran, S., Nigam, A and Wu, F.Y.: Artifact-centered operational modeling: Lessons Learned from Customer Engagements. In: IBM Systems Journal, vol. 46, issue 4 (2007)
5. Liu, R., Bhattacharya, K., and Wu, F.Y.: Modeling Business Contexture and Behavior Using Business Artifacts. In: Proceedings of the 19th International Conference on Advanced Information System Engineering CAiSE'07, Lecture Notes in Computer Science 4495, pp. 324-339 (2007)
6. Nigam, A. and Caswell, N.: *Business* Artifacts: An Appraoch to Operational Specification. In: IBM Systems Journal, vol. 42 No. 3, pp. 428-445 (2003)
7. Sukaviriya, N., Sinha, V., Ramachandra, T., Mani, S., and Stolze, M.: User-Centered Design and Business Process Modeling: Cross Road in Rapid Prototyping Tools. In: Proceedings of INTERACT 2007, pp. 165-178 (2007)
8. Unified Modeling Language (UML), Version 2.1.2, http://www.uml.org/
9. Business Process Execution Language for Web Services, http://www.ibm.com/developerworks/library/specification/ws-bpel/
10. COGNIam, www.pna-consulting.nl/
11. IDEF0: Function Modeling Method, http://www.idef.com/idef0.html
12. XPDL, http://www.wfmc.org/XPDL
13. Object Management Group's Information on Business Process Modeling Notations, http://www.bpmn.org/

14. Paterno, F., Mancini, C.: Model-based Design of Interactive Applications. In: ACM Intelligence Magazine Winter, pp. 26-37 (2000)
15. Paterno, F., and Santoro, C.: One Model, many Interfaces. In: Proceedings of 4th International Conference on Computer-Aided Design of user Interfaces CADUI 2002, pp. 143-154 (2002)
16. Paterno, F.: Tools for Task Modeling: Where We are, Where We are Headed. In: Proceedings of International Workshop on TAsk MOdels and DIAgrams for user interface design TAMODIA 2002, pp. 10-17 (2002)
17. Puerta, A. and Maulsby, D.: Management of Interface Design Knowledge in MOBI-D. In: Proceedings of 2nd International Conference on Intelligent User Interfaces, pp. 249-252 (1997)
18. Puerta, A., Cheng, E., Ou, T., and Min, J.: MOBILE : User-centered Interface Building. In: Proceedings of CHI 99: ACM Conference on Human Factors in Computing Systems, pp. 426-433 (1999)
19. Mayhew, D. The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design. 1st Edition. Morgan Kauffmann (1999)
20. Puerta, A., Micheletti, M. and Mak, A.: The UI Pilot: A Model-based Tool to Guide Early Interface Design. In: Proceedings of the ACM Intelligent User Interface 2005, pp. 215-222 (2005)
21. Nunes, N.J., and Cunha, J.F.: Towards a UML Profile for Interaction Design: the Wisdom Approach. In: Proceedings of 3rd International Conference on the Unified Modeling Language UML 2000, pp. 101-116 (2000)
22. Griffiths, T., Barclay, P.J., Paton, McKirdy, J., Paton, N.W., Gray, P.D., Kennedy, J., Cooper, R., Goble, C.A., West, A. and Smyth, M.: Teallach: A Model-Based User Interface Development Environment for Object Databases. In: Interacting with Computers, vol. 14 no. 1, pp. 31-68 (2001)
23. Griffiths, T., Barclay, P.J., Paton, McKirdy, J., Paton, N.W., Gray, P.D., Kennedy, J., Cooper, R., Goble, C.A., West, A. and Smyth, M.: Teallach: A Model-Based User Interface Development Environment for Object Databases. In: Interacting with Computers, vol. 14 no. 1, pp. 31-68 (2001)
24. Bouillon, L., Vanderdonckt, J., and Chow, K.C.: Flexible Re-engineering of Web Sties. In: Proceedings of ACM Conference on Intelligent User Interfaces IUI 2004, pp. 132-139 (2004)
25. de Baar, D., Foley, J.D., and Mullet, K.E: Coupling Application Design and User Interface Design. In: Proceedings of CHI 92: ACM Conference on Human Factors in Computing Systems, pp. 259-266 (1992)
26. Kim, W.C., and Foley, J.D.: Providing High-level Control and Expert Assistance in the User Interface Presentation Design. In: Proceedings of CHI 93: ACM Conference on Human Factors in Computing Systems, pp. 430-473 (1993)
27. Penicht, V.M.R., Paterno, F., Gallud, J.A. and Lozano, M.D.: Collaborative Social Structures and Task Modeling Integration. In: Proceedings of DSVIS 2006, pp. 67-80 (2006)
28. Sousa, K., Mendonca, H. and Vanderdonckt, J.: User Interface Derivation from Business Processes: A Model-Driven Approach for Organizational Engineering. In: Proceedings of ACM Symposium on Applied Computing, pp. 553-560 (2008)
29. Sousa, K., Mendonca, H. and Vanderdonckt, J.: Addressing the Impact of Business Process Changes on Software User Interfaces. In: Proceedings of the 3rd IEEE/IFIP International Workshop on Business-Driven IT Management BDIM 2008, pp. 11-20 (2008)
30. Sukaviriya, N., Sinha, V., Ramachandra, T., and Mani, S.: Model-Driven Approach for Managing Human Interface Design Life Cycle. In: Proceedings of the ACM/IEEE 10th Conference on Model Driven Engineering and Language Systems MoDELS 2007, pp. 226-240 (2007)