Ten Application Definition Best Practices

Improving application definition, before development, reduces risk while containing costs and driving top-line value.

iRise®
VISUALIZE. INNOVATE. DELIVER.™

This white paper describes application definition best practices that enable companies to cut rework costs, speed time to market and improve user adoption. These ten best practices improve the accuracy of application requirements, and improve the productivity of business analysts in defining applications, both of which drive tremendous value for organizations seeking to improve delivery of mission-critical business systems.

iRise ought to know. iRise has defined and helped to develop over 300 mission-critical applications for many of the largest corporations in the world. iRise is proud to share these best practices, refined through years of complex project experience.

Table of Contents:

Background

Rework is draining companies of their ability to reach the marketplace in a timely manner and innovate for competitive advantage. Rework, otherwise known as cost over-runs or change orders, is an endemic problem facing nearly every IT organization. According to Forrester Research, an estimated $103B was spent in the U.S. alone in 2004 on custom application development projects. A 2005 survey conducted by iRise and Decipher found that over 80% of organizations claim to finish projects within budget, yet almost three quarters (73%) budget for rework, thus, in effect, planning to fail. Almost a third set aside more than 25% in their budgets for these change orders, money that could be funneled directly into innovation rather than re-doing work that should have been completed the first time. Much of this waste is due to poorly communicated requirements.

Ultimately, rework costs companies the ability to get to market quickly. So while some companies are busy fixing applications, their competitors are busy capturing market share.

And when it comes t o business-critical portals and applications, it is essential that users be involved in defining how that application will function. Creating an application that users will not adopt is worse than not creating an application at all.

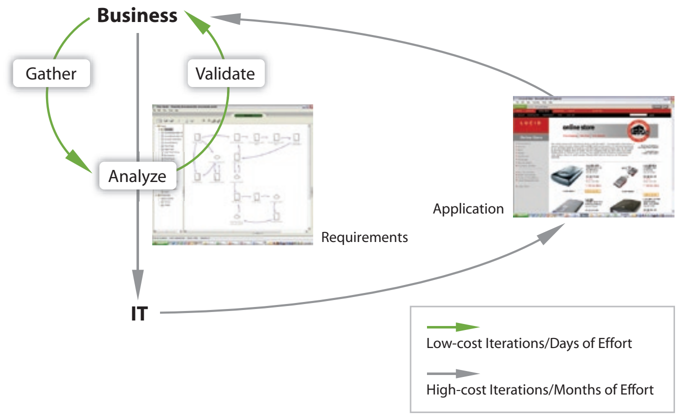

Business
Gather
Validate
Analyze
Requirements
Application
IT
Low-cost Iterations/Days of Effort
High-cost Iterations/Months of Effort

FIGURE 1 - *Allowing the business to define and validate the application (green arrows)–without tapping IT resources–dramatically improves the definition process.*

The answer to these costly, frustrating problems can be found in the creation of accurate requirements – before development. By allowing the business analyst to collaborate with stakeholders, users, architects, user experience designers and developers while the application is being shaped, all parties know exactly what will be built long before a single line of code is written. Figure 1 shows what can happen when iterations occur between the business and IT (gray arrows), which can result in months or years of work, and when iterations are handled by the business analyst (green arrows), which can shrink iterations to mere days or weeks. The purpose of implementing these best practices is to focus on these low cost iterations, and to improve processes so the cycle shown by the green arrows shrinks even smaller.

The best practices outlined in this paper show how to shrink the cost and burden of requirements definition. The benefits are astounding. By including the needs of all relevant stakeholders early on, not only does the cost of rework plummet, but the application is delivered faster and user adoption skyrockets. And by pulling developers from reworking their application, companies are able to apply those precious resources to more innovative, strategic projects.

Applying These Best Practices

Since each company, and each development team within each company, has their own way of completing projects, this paper does not attempt to map the application definition best practices to any particular application development process or the full SDLC. In fact, these best practices are laid out intentionally agnostic of any process so they can be applied to each companies'–and each departments'–own way of doing business.

These best practices fit into RUP just as easily as they fit into Agile Programming or any other methodology and are intended to assist the way applications are defined, progressing through the following:

1. Planning
2. Iterating
3. Validating

This three-step progression allows business analysts to think about their projects and contributors in terms of how the application should be defined, not built. Applications are defined first, built second. By following the ten tips outlined in this paper, whatever process or methodology that the team uses should progress efficiently and quickly.

Planning

It's common knowledge that planning is essential to keeping things on track and to keeping contributors engaged and efficient. However, with application definition as a specific task being relatively new to many, this paper introduces a few tips to help the process beyond common planning advice.

This section starts by introducing best practices in the planning stage to help companies and departments to improve consistency among team members, reduce the number of requirements to manage and improve the control of the management process. Sure, many more tips can be applied to the planning stage that are not described here, these are simply the top producers from many years of experience.

1

Step 1

Challenge: Team's skill sets are inconsistent

If getting people to agree on what to order for lunch is troublesome, getting them to agree on an application specification can feel downright impossible, especially if each person has unique backgrounds and skills. Inconsistencies can cause confusion and lead to time-consuming inefficiencies.

Solution: Define and employ accelerators

With a little training in software tools and templates, and the introduction of standard processes and standards, teams can be given commonalities to facilitate faster agreement. They're called accelerators because they enable the entire project team to drive toward a common goal using common tools and processes quickly and efficiently. Most companies and departments recognize the benefit of such tools, yet often times don't fully take advantage of them or employ them consistently. The most powerful accelerators that drive consistency among the team include:

- **Common templates**

Common templates and samples for specific deliverables give people a starting point so they don't "reinvent the wheel" every time they begin to define a use case or build a page.

- **Common tools**

Training people on common tools enables contributors to work freely without worrying about accessibility or file translation errors from one application to the next. For instance, standardizing on applications such iRise, Microsoft Office and Dreamweaver assure a seamless transfer of files from one person to the next without loss of fidelity or formatting.

- **Common processes**

By standardizing roles, activities and checkpoints, the project team will know exactly where they stand and what to expect at each stage of the process. And as new projects are undertaken, the contributors will have a deep understanding of these factors going into each new project, further encouraging rapid consensus.

- **Common standards**

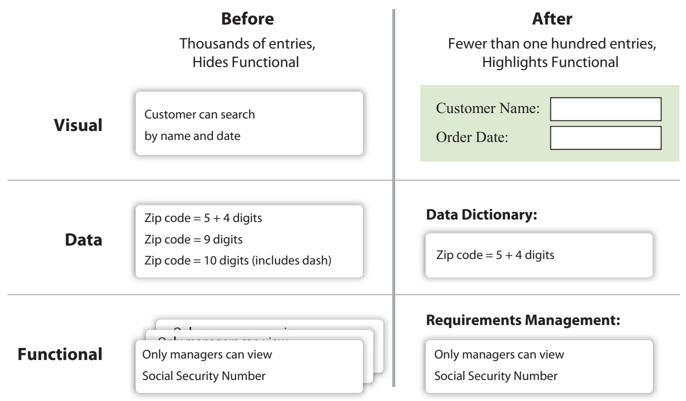
The best way to throw a project off-track is to introduce new standards along the way. It is best to identify and agree to a fixed set of standards for requirements, issues, naming conventions and change management so that no one is surprised with each new version. The less the team has to think about such standards, the more time they can spend defining the ideal application.

Step 2

Challenge: Too many requirements to efficiently manage

Requirements documents can easily balloon out of proportion. While many causes can contribute to this, some of the reasons that requirements get out of hand are: lack of simulation making it impossible to really see what's going on, requirements that are duplicated over and over for each use case and requirements management tools that are not properly implemented. Business analysts need to filter out the noise so stakeholders can focus on what's important.

For instance, including a zip code seems like a simple enough requirement. However, including a zip code is easy until one considers the three forms that any zip code can have: five digits, nine digits (five digits + four) and ten digits (five digits + four that includes the dash as a character). Such confusion can wreak havoc on the requirements document as it appears in multiple places that must be managed.



Before
Thousands of entries, Hides Functional
After
Fewer than one hundred entries, Highlights Functional
Visual
Customer can search by name and date
Customer Name:
Order Date:
Data
Zip code = 5 + 4 digits
Zip code = 9 digits
Zip code = 10 digits (includes dash)
Data Dictionary:
Zip code = 5 + 4 digits
Functional
Only managers can view Social Security Number
Requirements Management:
Only managers can view Social Security Number

FIGURE 2 - *Consider the merits of each requirement and tailor handling of it based on what it is.*

Solution: Consider requirement types and best management techniques for each

It is best to categorize requirements for better management, to employ a visual mechanism because a picture is worth a thousand words, and to reduce potentially thousands of requirements into a few hundred using a data dictionary.

To illustrate this recommendation, consider the example illustrated in Figure 2. Rather than duplicating a zip code requirement each time it appears throughout the requirements document, and being forced to carefully update each and every instance, it is best to show how it will look in a simulation or include it only once in the Data Dictionary (even a simple spreadsheet helps tremendously) or a requirements management tool. This way, the requirement can be changed once and the requirement will be reflected throughout. Likewise, rather than document in words how a login screen should look, a simulation, or at least a screen shot, can show what it will look like with a quick glance.

3

Step 3

Challenge: Requirements management process is overwhelming

Increasingly complex requirements require more and more effort to manage. Employing a tool to manage traceability can help, but complexity drives up the effort required to manage tracing. This increased effort of managing complex requirements can become such a burden that no one will even use the traceability tool.

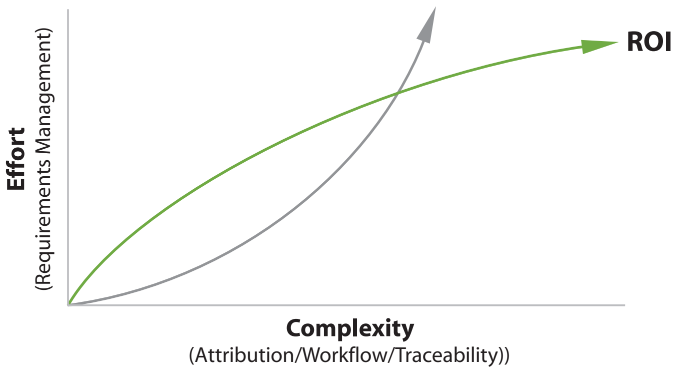


FIGURE 3 - *Balancing complexity with the effort required to manage it is a delicate act, and one which can reap tremendous rewards if handled properly.*

Solution: Create efficient requirements management plan when managing atomic requirements

The better companies and departments are at managing complex requirements, the greater return on investment (ROI) the company can realize from them. The key to managing complexity is in finding the simplest management process that produces a decent return while keeping the complexity low enough that it's not too overwhelming.

Eliciting

Next in the natural progression to improving application definition speed and accuracy is in eliciting requirements from the project team. While it is impossible to include every requirement that every contributor wants, business analysts can get very close by collaborating with the team, prioritizing all of the requirements and focusing on the most difficult ones first. It's a little like emergency-room triage, and the tips throughout the rest of this paper help to include as many requirements as possible without delaying the project. The following three tips help to keep the project on course while requirements-gathering continues.

Step 4

Challenge: Requirements are complete, but still ambiguous to developers

There's nothing more destructive to an application definition than ambiguity. The potential pitfalls are many: requirements are perfectly clear to stakeholders and users as written, but developers still build them incorrectly; or the requirements are complete, but developers still don't get it. While simulation is the best means for overcoming ambiguity, additional methods can further improve the odds against this potential project killer.

One of the most common causes for ambiguity is when actions are based on rules rather than on behaviors. Rules are fine for determining system-level requirements. But when it comes to determining the functionality of applications that rely on users and user input, rules are not an effective means for eliminating ambiguity.

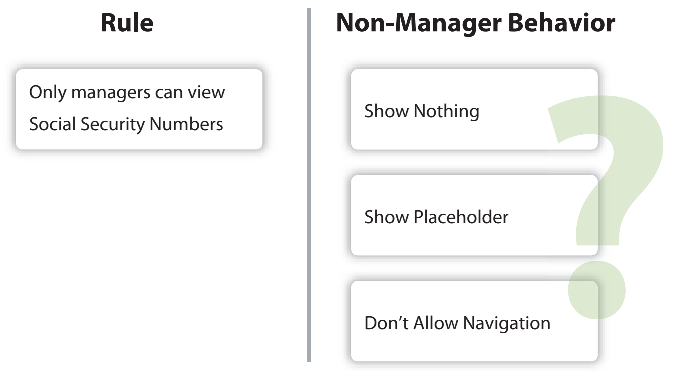


FIGURE 4 - *Focusing on user behavior reduces ambiguity by forcing the application definition team to think about the application in terms of how it will be used, not just how it will function.*

Solution: Document requirements as behavior where possible, i.e. if (action) then (response)

Machines are predictable, people are not. That's why rules should be used for system requirements and behaviors should be used to govern user requirements. Always scrutinize the expected behavior of any rules that are captured during the elicitation process, and document any potentially ambiguous rules as behaviors (i.e. how should the system respond to either the rule being met or broken?) This approach will ensure that later, when the application is being developed and tested, it will be very close to what users need.

One way to correct the situation illustrated in Figure 4 would be to simply change the rule to a behavior that responds with information that serves all potential use cases, such as:

```
When non-managers attempt to view a Social
Security Number, present “Social Security
Number not accessible”
```

This assumes the inevitable user activity and solves a complex set of ambiguities with a single line.

5

Step 5

Challenge: Requirements can’t be implemented as defined

A surefire way to delay rollout of an application is to lock the definition team in a room for months and months writing the requirements document, then throwing it down the hall to the IT department to build. Square wheels on a car work better. The most common reason for this behavior is that today’s tools simply do not allow for on-the-fly collaboration. People can’t sit in a room together and iterate on a giant text document. It’s simply not feasible. So, typically, the business refines the requirements down to a point where they feel comfortable with the specification and they send it to IT. Or the definition team simply runs out of time and ‘accepts’ requirements they know to be incomplete or just plain wrong.

Solution: Employ an efficient method of communication to allow construction/transition resources to participate and provide guidance early and often

The sooner and more often user experience (UX) designers, quality assurance (QA) and architects are able to contribute to and comment on an

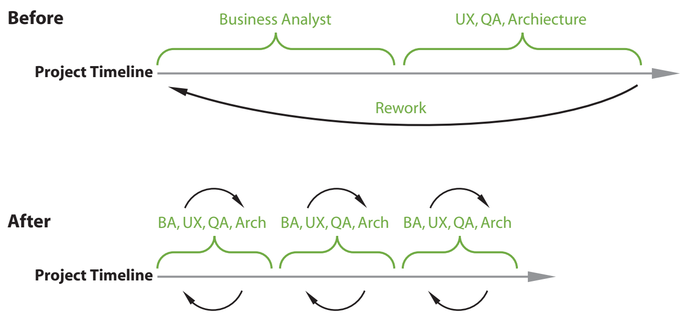


Figure 5 - *Shrink the overall timeline by including user experience, quality assurance, and architects more often during the definition phase.*

application, the sooner the final specification—and the final application—will be complete.

As Figure 5 shows, by including the entire team in the definition, using tools that allow real-time collaboration, the requirements go through more, but shorter, iterations. These shorter iterations produce a much tighter requirements document in a shorter amount of time and developers face a lower likelihood of having to redo their work. The business analyst is responsible for creating the specification, UX, QA, architects provide minimal but highly valuable input, and, when they're all done, developers build it.

Step 6

Challenge: Analysts focus on simple requirements and not on difficult requirements

When reviewing the requirements of a large-scale application, it is easy to lose sight of what's critical. Line after line and page after page of detailed requirements can numb even the savviest business analyst, let alone the most interested stakeholder. The sheer number of requirements makes it difficult to efficiently clarify the most important ones.

Solution: Iterate through increasing levels of detail and focus scope only on areas of ambiguity

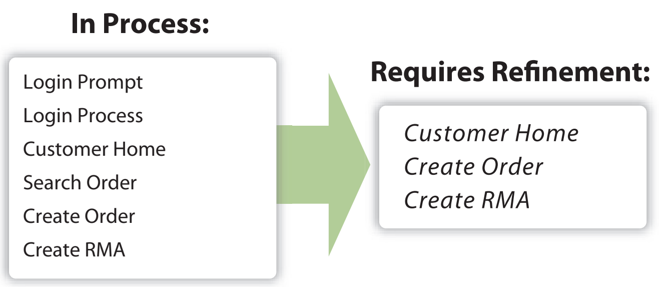


Figure 6 - *Singling out ambiguous requirements encourages the team to focus on requirements that truly need clarification.*

The best way to reduce the total number of requirements, and encourage the team to focus on areas of ambiguity, is to single out requirements that are ambiguous while leaving relatively clear requirements alone. As the project proceeds, the team can return to those requirements that still have a level of ambiguity associated with them and iterate through increasing detail as time allows. Ambiguous requirements get the attention they deserve without overwhelming the team. As time permits, the definition team can revisit the ambiguous requirements until a level of refinement is acceptable to the development team.

Validating

Sending an application specification for development can test the nerve of even the steeliest business analyst. The defined application will either quickly spark the next wave of corporate profit, or ignite the wrath of angry developers. Before sending any specification to developers, it has to be complete and unambiguous. In order to assure a complete specification that will capitalize on rapidly changing market demands, it's very helpful to acquire approval from all interested parties, minimize iteration time, clarify stakeholder meaning and trace requirements based on their level of detail.

Step 7

Challenge: Requirements are clear to IT team, but not to Stakeholders

While stakeholders can visualize corporate profit from final applications, they don't always see the same level of detail and functional intricacies that developers do while they're deciding how it will work. For instance, stakeholders may be used to seeing applications hosted on a mainframe with a function-key paradigm so when a web-based, mouse-driven application appears they may not be able to provide valuable contributions and improvements. They expect to see one thing, and the specification describes another.

Solution: Consider project team's experience and/or familiarity of domain and technology and proceed accordingly

Business analysts should always know as much as possible about the team prior to starting the next project. How knowledgeable the team is affects the level of detail that will be required to accurately describe the application. If they have little experience with the proposed technology, the specification will have to be very detailed, walking them through the application every step of the way. If, however, they have a lot of experience with the technology, the specification probably need not be exacting in descriptions of pages, interactions and processes and focus instead on what's changing in the proposed application or business process.

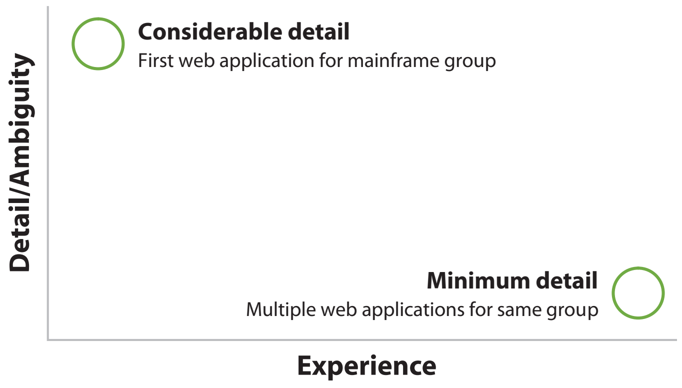


FIGURE 7 - *Consider the team's knowledge before defining the next application.*

Recognizing and accommodating stakeholder knowledge of the technology and domain is critical to providing a specification that offers the appropriate amount of detail. Not too much, but just enough to acquire validation is the key to a fast approval process.

8

Step 8

Challenge: Requirements review iterations are too time-consuming

"Laborious", "time-consuming" and "dull" are some of the terms that have been used to describe stakeholder review meetings. It's not that the subject is boring, it's that the process taxes even the best business person. And they're difficult to follow in part, because, traditionally, they've been based on text that people simply cannot digest while interacting with others. People either read or converse, they can't do both at the same time. And since text is not a medium that facilitates collaboration, one person works on the requirements, then passes it to the next, and they pass it back when they're done, and so on. Not only does each iteration take time, time is wasted because only one person can work on it at a time.

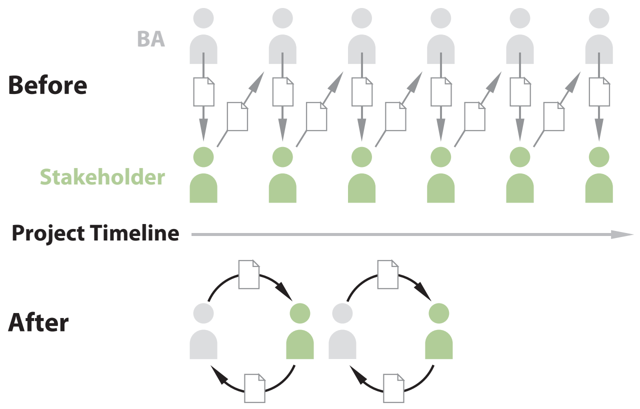


Figure 8 - *Modify-on-the-fly tools are essential in facilitating real-time collaboration to shrink timelines.*

Solution: Eliminate review iterations using modify-on-the-fly tools

Rather than attempting to interpret text into screens shots, stakeholders need to see exactly what's going to be built so they can comment on and improve it right there in meetings. By showing the specification and iterating the design while everyone is in the room, stakeholder meetings are far more fruitful.

Application simulations take the dread out of stakeholder review meetings by showing what the application will look like. Stakeholders can play with the application long before IT ever gets involved so they can provide feedback about the experience as well as how it looks and functions. Even HTML or static screen shots are better than text for speeding this process. The key here is to utilize any tools that allow real-time collaboration.

9

Step 9

Challenge: Stakeholders don't mean what they say

Similar in context to step 8, text is a poor way to present an application's requirements. Text leads to confusion about what is going to be built. So often, text specifications are a developer's nightmare. Simply put, text should never be the sole context upon which developers should rely for creating final applications.

Solution: Validate requirements by presenting them in the context of visual artifact

Simulation is essential in translating stakeholder meaning into an accurate user interface. Only through seeing an application's interface, and the impact from data interactions and process flows, can one truly comprehend intended look and feel.

FIGURE 9 - *Text should never be the sole means for conveying requirements.*

The reason this is so important can be demonstrated in describing the branding guidelines that the application must follow. Branding guidelines are often based on perception, which can easily be misinterpreted. Larger companies often publish detailed branding guides. Yet, those guides sometimes contain more than one specification – one for customer sites, another for intranets, and so on. Without a visual specification, the development team is required to interpret the design into their site. Interpretation leads to ambiguity, which should be avoided whenever possible.

10

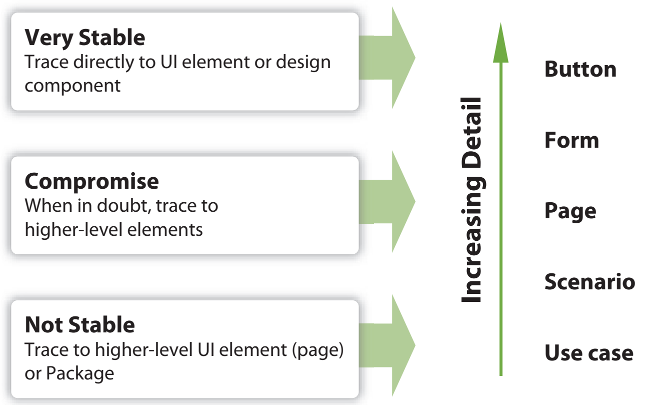
Step 10

Challenge: Too much rework in managing traceability

Requirements traceability can be a valuable tool, but when the traceability is assigned at a very detailed level, rework associated with maintaining the relationship can skyrocket as the project changes. Tracing to a highly detailed level can require changes to that relationship over and over, easily gobbling up 70% of a business analyst's time. An example might be a requirement traced to an input field; as the form and the form's supporting requirements change, the trace relationship needs to be managed, causing a cascade of rework.

Solution: Trace requirements at a level of detail that reflect stability of requirements and focus on minimizing rework/management

The key to using traceability effectively lies in evaluating each requirement for tracing suitability and adjusting accordingly. The stability of the requirement, as well as the stability of the associated component, is important is because detailed traceability relationships improve accuracy. But since there is a rework tax attributed to maintaining the relationship, it's important to only use detailed traces when stability is high.



Very Stable
Trace directly to UI element or design component
Compromise
When in doubt, trace to higher-level elements
Not Stable
Trace to higher-level UI element (page) or Package
Increasing Detail
Button
Form
Page
Scenario
Use case

FIGURE 10 - *A requirement's stability is the first criteria to consider when utilizing traceability.*

If the requirement is very stable, business analysts can trace to a detailed level such as a button or text widget in UI traced in a page or a specific step in a use case. But if it's unstable, detailed relationships should not be made because they have to be constantly managed. In such cases, business analysts can trace to higher level components, to pages or use cases or use case packages. If stability is not known, trace to higher elements; it can always be cleaned up after it is stable.

Conclusion

Applying best practices in application definition can reap tremendous rewards. While every wrinkle in every project can't be ironed out, this paper describes many ways to boost productivity and maximize efficiencies, which ultimately improves the bottom line. By improving the accuracy of specifications through application definition best practices, rework is cut and time to market and user adoption are improved. iRise hopes this paper becomes a valued reference for future application definition projects.

To view the online demo of the iRise platform, please go to:

http://www.irise.com/demo

Call iRise for a product demonstration and be sure to ask about our free proof of concept.

iRise®
VISUALIZE. INNOVATE. DELIVER.™

1-800-556-0399 www.irise.com
2321 Rosecrans Avenue, Suite 4200
El Segundo, CA 90245