Seagate Software

Implementing Seagate Holos

A Guide to Rapid Application Delivery

Contents

Introduction

"It is common sense to take a method and try it. If it fails admit it frankly and try another. But above all, try something."

Franklin D. Roosevelt

In the fast moving world of business intelligence, the only thing that is certain about the business requirements is that they will change, often before the solution has been implemented. Usually the very act of implementing a decision support system will change the way in which the users operate, thereby altering the requirement. This paper sets out the principles for delivering successful applications in Holos which deliver real benefits to the users, on time and within budget.

Given the range and scale of Holos applications, there is no single project methodology or approach that can be applied in order to guarantee the successful use of Holos. Holos is applicable throughout an organisation as a Business Intelligence solution, and can provide long term business benefits over many years. Many applications are complex, enterprise wide with many hundreds of users. Many customers have their own in-house methods for Holos that they have used very successfully. On the other hand, some Holos projects can be very small, single person systems developed in a few hours.

This document concentrates on our *'Rapid Application Delivery'* approach. Typically, we are seeking to deliver the first few applications of Holos within an organisation very quickly in order to gain some benefits from the initial purchase of Holos. Rapid Application Development or RAD is not a new concept, but the unique tool-kit approach within Holos allows the approach to be exploited to the full and assists the *delivery* of real business benefits. RAD with Holos is a way of ensuring that the project concentrates on delivering 'eighty percent of the business benefits' for 'twenty percent of the effort'.

RAD is clearly not the only approach and we are always very happy to discuss peoples' experiences with implementing Holos.

Why do I need Seagate Holos to do this?

Holos provides an application tool-kit environment that allows a multi-disciplined team to deliver applications quickly and efficiently. Tools such as the worksheet and the report designer allow a user to quickly build a reporting application. Modelling and what-if functionality can also be added to this via interactive services through the report designer. There are tools at the lower level such as the SQL Select Builder and a supporting Holos fourth generation language that enable an application developer to deliver an unlimited range of user applications.

Here are some reasons why Holos is particularly productive in the RAD environment:

- the development stage itself takes very little time, because 'programming' in Holos is very productive and fast. This encourages an iterative approach and enables small applications to be built very quickly and modified as necessary.
- The graphical user interface of Holos lends itself to a high degree of user involvement, and it becomes very wasteful to invest too much time in specifying the on-screen application in too much detail. It is important to set some standards for the interface to ensure consistency across the applications of Holos (such as colour scheme and button styles), and then allow users to build screens if they wish rather than specify their requirements for someone else to develop.
- Holos itself is advancing rapidly with each version, therefore the users will gain from our ongoing development of the underlying Holos product as well as enabling more and more sophisticated solutions to be delivered. It will be important to ensure that you design applications so that they can take advantage of future versions of Holos and we can help with this.

- Holos is 'data-driven' which means it is possible to design a system using Holos where all of the information about the design of the system is based on the data itself rather than being embedded in the Holos application in some way. Therefore any changes to the 'shape' of the data such as the addition of new products or revised reporting hierarchies can be immediately reflected on the Holos screens with no maintenance overheads. It is essential to design modern and changing business applications in this way.
- Holos integrates very closely to an underlying relational data warehouse, enabling dynamic two way links to be set up between the Holos application and the data/meta-data. There is no necessity to replicate data, simply to map directly onto the existing data, enabling users to report, model and input in a multi-dimensional environment.
- Holos is scaleable from single user models similar to those created through spreadsheets (and just as easily), through to large-scale enterprise wide business applications. The range and solutions can grow easily to meet the business need without additional effort.
- Holos is a long term solution - we have customers who have used Holos for many years, since its inception. It is adaptable to the changing business requirements and the Holos product is moving forward rapidly in terms of its own functionality, yet retaining upwards compatibility from one version to the next. This track record provides a confidence in the future of Holos that permits the application developer to take a longer term view of the solutions being implemented.
- Holos can be applied across the entire business spectrum, including finance, sales and marketing, manufacturing, distribution, personnel, project/programme management. This enables a broad base of usage to be encouraged, thus reducing the overall cost of ownership and enhancing the benefits to the business.
- Holos is a client-server application, not solely PC based. It takes advantage of the best of both worlds and enables the implementers of Holos applications to distribute the solution very widely in the organisation for very little effort.

Seagate Holos projects

'Non RAD systems' - Small scale developments

Holos can be used to develop a relatively small application that represents a particular business situation for one user. In this situation, the user will be reluctant to undertake what appears to be the large overhead of any methodology or standard approach to development, even a RAD approach. This situation is similar to the typical developments that occur with spreadsheets, and it is perfectly possible to make very good use of all of the Holos features with this approach.

Very quickly, however, the application grows and more importantly becomes multi-user and there are thoughts of introducing some consistency in the

development approach of the different users involved and some means of accessing and sharing the underlying data.

There are no ‘brick walls’ with Holos and it is not necessary to throw away anything already developed, since all can be extended and shared without limits. This flexibility provides enormous benefits, and brings with it the opportunity to organise the deployment of Holos within the organisation to suit the structure of business in place.

For example, many customers have IS or IT departments that are responsible for providing the underlying data infrastructure using Holos, and then enable their user departments to organise and develop their own business applications. The shared computer resources and the shared data are organised by IT, the individual user applications are either developed by the user or someone local to the user’s business function. In effect IT provide the building blocks for the users to put together business applications as they wish.

These small scale user developments can evolve into RAD projects where the actual development is undertaken by users.

‘Non RAD systems’ - Large scale developments

Clearly Holos can be used for large scale, enterprise-wide implementation of a single application or a suite of applications and the RAD approach may not always be appropriate. Other methodologies can be applied just as well.

It is always a good idea to involve users in the definition and development of the screen based aspects of the system, and never a good idea to try and define these on paper in too much detail. Much of the interactive functionality of the application can be specified through workshops attended by the designers of the system and the users. The results of these workshops can be captured and formally signed-off if required.

Much of the underlying data capture and base data manipulation can be designed and developed by preparing requirements, technical and program specifications. Appropriate sign-offs and formal test procedures can be incorporated in the normal way.

Even if the project is clearly not RAD, it is often a good idea to get some initial experience with Holos by undertaking a smaller, RAD project first. This provides invaluable experience and gives a better view of the estimation process for larger projects.

The Seagate Holos 'Rapid Application Delivery' approach

The traditional approach to business systems development encourages a project scope to be defined and the success or failure of the project to be measured by the timeliness and cost of the end results. Most projects are sub-divided into segments, or modules or some form of staged-deliverables that enable progress to be measured and corrective action taken if there is unacceptable slippage or cost over-run.

The Holos RAD approach looks at this from a completely different perspective. We are more concerned with *why* business systems are being developed rather than *what* is being developed. The intent of any new system is to deliver some greater benefit to the business, either through enhanced revenues, improved customer satisfaction, greater efficiencies or whatever. It is these benefits that are the real deliverables rather than the stage by stage deliverables of the development project.

Recommended stages in a Seagate Holos RAD project

The section which follows explains the roadmap for a successful RAD implementation. Figure 1 provides a diagrammatic view of the RADelivery approach with each stage described in some detail below.

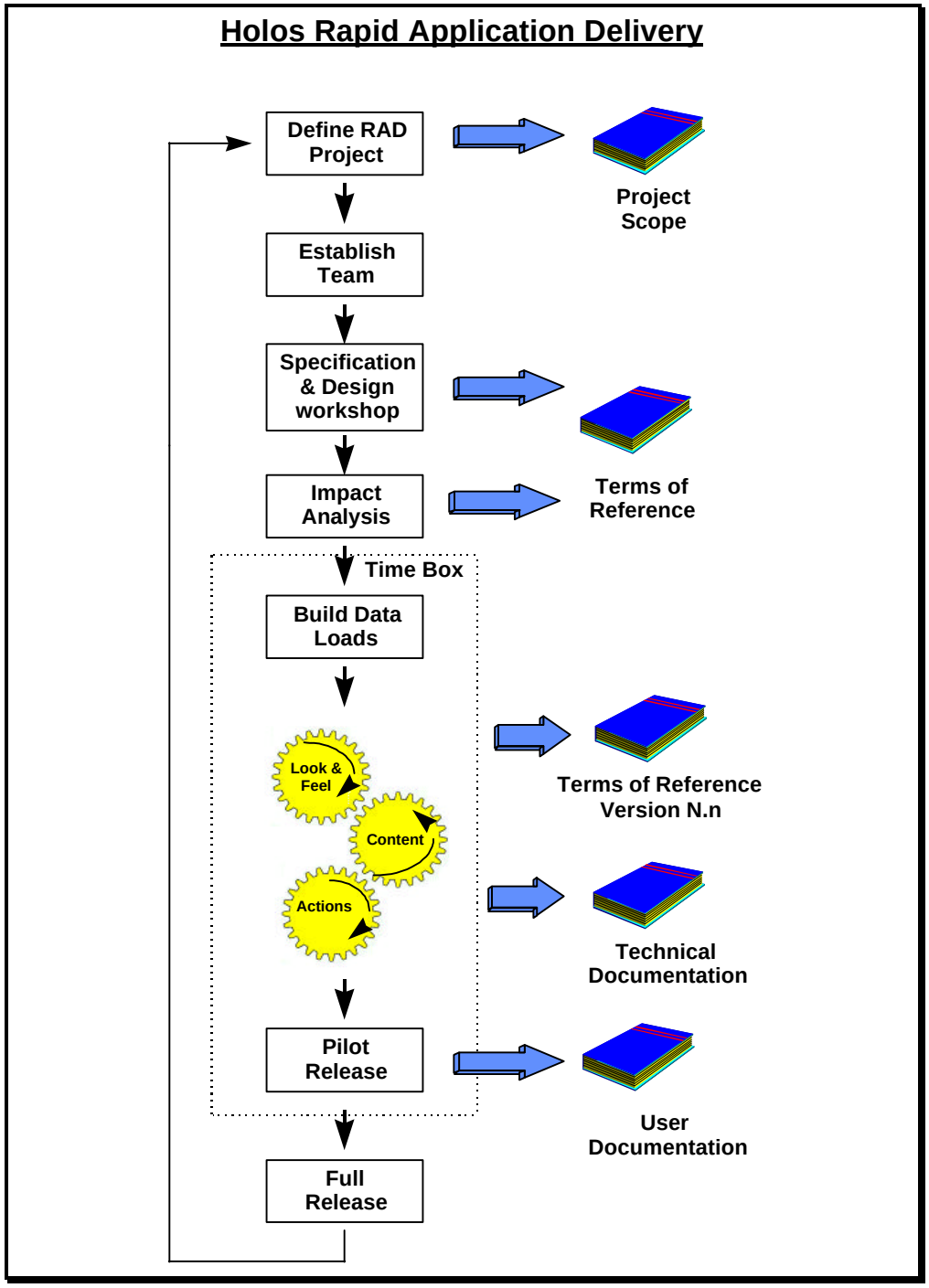

Figure 1

Define the Scope

Firstly it is important to identify whether it is possible for the project to be undertaken with the RAD approach. As already explained, Holos can be applied to a wide range of business applications and not all of them can simply be labelled as 'rapid deliveries'. In order to qualify for RAD:

- it should be possible to define the scope of the application such that it is going to be possible to develop in a few weeks with no more than two or three people. It is unrealistic to expect to deliver a large-scale, enterprise-wide application targeted at hundreds of users under RAD.
- Users must be committed and available for the project - not necessarily full time, but an agreed level of commitment.
- Data for the application should be available already, either in an existing relational database, or in a clearly defined and agreed source.

Once the project has been identified as appropriate for a RAD methodology it is necessary to take control by:

- Identify the major players and get their buy in.
- Define and prioritise the deliverables

The goals and objectives of the project should be established at the outset, including a view of the business benefits that are being sought. E.g.: 'the analysis of sales by product and commodity group to enable exceptional performance to be identified.' These objectives should be written down in a project scope document that is the output from this stage of the project. This document can then be incorporated in the 'Terms of Reference' document.

Establishing a Seagate Holos RAD team

Set up a Holos RAD team of users and developers that you can completely empower to deliver the business application. It is very important to get the right team, with the right skills and the right amount of input from users. Keep the overall team to no more than six people, consisting of two or three Holos application designers/developers combined with appropriate representation from the business users.

Suitable skills for a Holos application designer/developer are:

- Database design and implementation.
- Ability to program, preferably in a 4GL or modern application language
- Good business understanding and strong analysis skills
- Close links with the user community

The representatives from the user community should have sufficient knowledge to be able to define the requirements and describe the business in some detail, and also carry the authority to agree to the overall functionality. It is likely that you will only be able to get part-time commitment from them. The right users will probably be those who don't really have time to spend on a project like this and will therefore need persuading to contribute a minimum amount of their time (such as a day a week or a few hours per day). Be suspicious of users who volunteer their services full time - they're probably not the right people!

As part of the above team, if it is your first Holos RAD project, it is advisable to include some consultancy skill from Holistic Systems. This will be particularly helpful when designing the Holos application and Holos data structures. It is possible to set up Holos training and design workshops outside of the RAD project itself in order to provide the development team with the appropriate Holos skills.

Once the team has been formed it is important that they are given the freedom to become empowered. Without that independence bureaucracy will creep into the process and slow down the development and stunt any innovation. Some important features of a RAD team are:

- Minimal project reporting and control procedures.
- Focus on the business objectives. Keep the definition of requirements short and simple in the first instance. Cut out any 'nice to have' features that do not deliver any real benefits.
- RAD project teams are significantly smaller than those of traditional teams.
- Give them a timebox of six or eight weeks and trust them to deliver some real business benefit. The idea of a timebox is to define the length of the project rather than the content. Make the content fit the timescales for delivery.

The Terms of Reference

Having established there is a RAD project, and decided upon the project team as described above, the next stage is to produce a *terms of reference* for the development. The *terms of reference* acts as a project charter and ensures that everybody associated with the project is clear about the goals and objectives and what their respective roles are in ensuring that they happen.

The *terms of reference* should be treated very much as a iterative working document which is updated each time the requirements are refined or new issues and risks are introduced to the project.

The best way to produce this document is to undertake a 'Specification and Design Workshop'. This will normally be no more than a day and the entire team should be present for the workshop. The agenda and terms of reference document should encompass:

- Confirmation of the scope of the application.
- An agreement of the functionality to be incorporated, including number and complexity of reports, definition of rules and calculations, description of any modelling/forecasting required, and definition of any specific functions required.
- A statement of the source of the data and clear understanding of who is responsible for its provision and quality.

- A summary of the technical environment, which machine is it going to run on etc. Undertaking a project in such a short time-scale can have an impact on other areas of the business, including in particular the provision of a suitable client-server technical infrastructure for Holos. Delivery of hardware, new disk drives, PCs, network cabling, etc. should be monitored carefully and any delays raised very quickly as issues by the project team. Implementation is the driving force to Holos RAD projects and so all aspects need to be considered.
- A definition of the roles and responsibilities within the team, including a commitment of how many days per week or hours per day that users will be involved. This could even get to the level of agreeing specific time-slots each day.
- Contingency Scope which should include all areas of functionality which are only to be included within the timebox should time allow. Otherwise these should form the foundation for the next timebox.
- A statement of any specific exclusions - Holos features or business data that the application will definitely not include. For example some users may interpret ‘sales data’ to include price, margin, etc. If this is not the case then say so.
- A milestone plan for the implementation.
- Any issues and risks to the project. This should be updated with each iteration of the document.

If the workshop and subsequent estimation process identify a project longer than about eight to ten weeks, it is important to de-scope the application, with a review of the required functionality against the business benefits. It is important to concentrate on the core business need, rather than be tempted to extend the functionality with ‘bells and whistles’.

Risk/Impact Analysis

All business systems change and recognising this is essential for the successful implementation of a Holos application. When you believe you have a first cut design for the Holos application as part of the workshop, it is important to undertake an ‘risk analysis’ on the design you have arrived at.

Some examples:
What would the impact of a new product or business unit be?
How will we cater for next year’s result?
Isn’t it likely that the user will require a time-series chart at this point?
What happens if we want to analyse the results by customer? etc.

The risk analysis stage is often forgotten, but it is sometimes very easy to arrive at a few questions that are actually quite likely to happen and can have a significant impact on the design of the Holos application.

The Milestone Plan

The project team, consisting of users and developers, is empowered to deliver the final application. There should be as few project controls as possible, concentrating on an effective and speedy resolution of any issues that arise that could affect its success. There are a number of standard milestones that can be defined in the overall project plan:

- Completion of the 'Specification and Design workshop'.
- Distribution of the Terms of Reference
- Impact analysis completed
- Data available and load routines written
- Agreed detailed definition of rules/calculations
- Completion of first prototype
- Completion of second prototype
- Pilot release
- Performance and quality review
- Full release

Don't concentrate too much on individual stages, program specs, system tests and so on. The project activities must be undertaken, but in a Holos RAD approach, it is not important to measure their cost. Don't review individual project members based on their productivity or whether they are ahead/behind their plan. You may feel that this leads to a lack of control, however because of the focused nature of RAD and the short timescales, the opposite is true.

The Development Timebox

Timeboxing is the mechanism that is use to control the progress of a project. The project is split into smaller packages, called timeboxes, with each timebox planning to deliver certain functionality within pre-determined timescales. The difference which this approach bring to the table is the emphasis on delivering on time. It is better to deliver something which is 80% complete on time (and provides a business benefit) rather than something which is 100% complete, but six months late.

Feeding into a timebox are:

- The *Terms of Reference* document, as discussed earlier, is treated as an iterative working document which is updated each time the requirements are refined or new issues and risks are introduced to the project.
- Data Loads
- *Prototyping* is an iterative approach to designing and building a business application.

- Holos
- Documentation and development *Standards*. During the course of the project documentation will be developed just as if it was another element of the code.
- Workshops

Within the timebox a number of prototypes would be build. The number of prototypes is not fixed - they should be used as a way of clarifying the purpose and functionality of the system. This is meant to be an iterative approach. Prototypes can be an effective way of ensuring that users are fully involved in the development of the application. For example, the first prototype can be a mock-up of the application that can be used to derive a consistent and acceptable 'look and feel' and help to define more detailed reporting requirements. Always use real data (remember that RAD projects will have the data available at the outset), since prototyping with randomised data can cause confusion and misrepresent the style of the reports. If you want to be really adventurous, why not let the users build the prototypes themselves?

There are three components to prototype:

- 'Look and feel' defines the way in which the application appears, how many desktops there will be, the layout of the screens, the colour scheme, fonts, button styles and locations, and so on.
- 'Content' defines the content of the application and each screen and paper report. This will include the type and content of each table and chart, the 'sectioning' of the report and the contents of any other screen objects.
- 'Actions' defines what will happen during the process of the application; what will happen when buttons are clicked, fields selected etc.

Testing & Preview Pilot

Having completed development, there should be a final testing phase. There will have been ongoing user acceptance testing throughout the project; this final testing concentrates on:

- the quality of data - is it adding up correctly, do the numbers reflect what they should?
- the functionality - does it meet the terms of reference?

This final testing can take the form of a 'pre-production pilot' that is put out to a wider audience than the project team itself and is used to finalise any details and ensure that full production system is well-received.

Documentation & Training

Documentation should be developed during the course of the project and not as a final stage. This will reduce the impact of developing documentation to a minimum. Much of the technical documentation can be automated through

Holos, by ensuring standard headers to objects are maintained throughout. User documentation will often be incorporated in Help screens or prompt cards, etc. The implementation phase should include a number of demonstrations and training workshops for the users, usually undertaken by the user representatives from the project team.

Receive accolades

It is very important to recognise success and there should be some form of celebration at the end of any RAD project. Tell people how well it went, what the benefits of the new system are, how happy the users are, how much better life is than it was before this system was available. Make a name for yourself with its success; maybe it's a new initiative in your organisation and there has never been this sort of application delivered so quickly. If so, then shout about it.

A successful application that is providing users with useful information and functionality will quickly require enhancements. The requirements will be changing and growing. Suitable plans should be drawn up to incorporate new requests, spawning new projects.

In this way, Holos applications delivered through a RAD approach continually evolve as the business needs change.

Summary of do's and don'ts for RAD

- Make sure you get the **right** users involved
- Get the level of project control right
- Examine skills of development team honestly and do something about any weaknesses through additional training
- Don't undertake any training too early or for the sake of it - do it in conjunction with the overall project plan.
- Keep it simple - question the true business requirement very hard
- Don't try and develop the complete system before showing it to anyone.
- Make it easy to access and use for the users - encourage them into it.
- Don't over-commit