Reclaiming your legacy applications through pragmatic application modernization

By

Marinus van Sandwyk
www.linkedin.com/in/mbogo

Discover how major companies re-committed their organizations strategically to IBM i, DB2, and RPG by modernizing their legacy applications rather than replacing them.

Find out the way to determine how much value remains in your heritage applications, whether they have a future, and what realistic approaches are available for you to leverage and reclaim them.

Get expert advice to present your database as “modern” to the new generation of SQL-literate managers.

Learn how to “sanitize” your database from years of neglect in a non-disruptive fashion and why and how to enrich your metadata.

Discover how to migrate your DDS-defined database to native DDL definitions in a non-disruptive fashion and why you should move to DDL and the SQE engine on DB2.

Leave this session with the information you need to get your CEO/CFO to support you on modernizing your heritage application and to leverage DB2 for IBM i and gaining competitive advantage.

1. Question – how much value is there in your legacy systems?

The key questions that you and your organization need to ask are:
How much value does your legacy system deliver to your company and your customers?
What is the cumulative value of the intellectual capital invested in your application? (This is usually much more than one might initially think and is worth quantifying).
What competitive advantage does your current system provide?

Often business will respond that there is little to no value in their legacy application. If this is true, then why is the company successful??
In every single instance where we have been involved internationally, the business leaders have quickly admitted that while the functionality delivered by their legacy LOB (Line-of-Business) application was of significant value, this was negated by severe operational constraints, the most significant of these being AGILITY.
If our experience is anything to go by, there remains MASSIVE value in your "legacy" applications and this can be leveraged relatively easily for dramatic results and benefit to your company. It is entirely feasible to extend the life of these applications for another 10 to 20 years, whilst remaining competitive!!

2. Legacy vs Heritage

The use of the word "legacy" in IT as something "bad" is something with which I truly grapple. "Legacy" in ANY other context is something valuable and good. In our opinion, it is high time that we as IT stand up and challenge this notion. There is ENORMOUS value in our legacy, provided that we remove the constraints or shackles. As a result, we prefer referring to our "heritage" and highlighting to business how this heritage has made us successful.

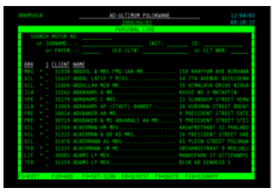
3. IBM i - a very powerful vehicle.

The fact of the matter is that we have the best commercial online transaction processing platform in the history of computing, yet our users quite often have a different perception.

4. Why, then, is your system perceived as "legacy"?

The old adage of "perception is reality" is accurate here. Regardless of what we know or believe, our users perceive our "top of the line" sports vehicle as an old, antiquated, and underpowered vintage model.
It is important that we study the reasons for this and that we correct these perceptions pro-actively. In our experience, the following facts contribute to this:

5. UI/UX – Rendering to delivery channel



The first problem is our User Interface and User Experience because it is the most visible part of our application. It is especially problematical if we want to expose this enormous functionality to the generation Y (and soon generation Z) consumers.
We believe that the way in which the DB2 database presents itself to the new generation of SQL-literate business leaders, hurts us a great deal - much more than we realise. We refer here to the 6 – 8 character field names and the 6 – 10 character file names, which we programmers have historically employed to name our database constructs and elements. These SQL-"savvy" users take one look at our database and immediately develop an intense dislike for the system we love, as they cannot

relate to our naming conventions and want to deal and interact with the database using familiar terminology, not "techno speak". These users (quite often the ones calling the shots) WILL try and get rid of "this remnant from the dark ages" as a matter of priority, unless WE change the game…

6. Old style monolithic code

Having said this, however, we believe that the biggest issue, and the one giving the most trouble, is hidden to a large extent from cursory review: the way we used to develop systems in the 1980's and 1990's! We used to develop these huge "structured" monolithic programs, as that was the programming model of the time. As a "rule of thumb", 80% of the lines of code in these monoliths dealt with database (entity) relationships and database validations. Additionally, there was little to no separation of function, which made maintenance increasingly problematic. The way in which this presented itself, was massive maintenance backlogs and frustrated users.

7. Why the reluctance to modernize?

Probably one of the strongest features of our platform has become our "Achilles Heel". Because the platform is so forgiving and has just kept on running our antiquated code, we began living in a "fool's paradise" – believing that our applications and systems were not "broken", so there was no need to fix them.
Therein lies the problem. We seldom or, more accurately, never improved our applications by introducing newer coding techniques, leveraging new enhancements to the language, operating system and database and improving the maintainability of the applications, because the system just kept on running.
Compare this for instance to any other mainstream environment. If Microsoft announces a new release of .Net, you have to REDEVELOP your application. This "head in the sand" attitude of ours, has come back to haunt us.

8. Is there a solution to this problem?

ABSOLUTELY - and you will be surprised how simple, natural and logical this is, and without a major upheaval. This presentation will study this process, which we call "Strategic Modernization".

9. Two modernization case studies

We will explain this approach by way of two case studies. Both of these case studies are currently being formalised by IBM Systems Magazine. We believe that by using real life examples and the discussing the results, you will relate better to the major opportunity presented to us all.

10. Cape Gate

Cape Gate is one of the Top 3 Steel Producers in South Africa. It is running a HEAVILY customized old System/38 based version of the JBA ERP application dating back to 1986. Due to the level of customization, they bought the source code in 1986 and have been self-sufficient and self-reliant for any required enhancements and changes. Due to the level of customization, they believed that new releases of the ERP application were not likely to benefit them. They run one of the top RPG development shops in South Africa and are currently on IBM i Version 6.1 with an upgrade to version 7.1 being planned. The order entry module, for instance, has been essentially redeveloped over the years from the ground up, providing Cape Gate with significant competitive advantage. Their own investment into their system exceeds 150 man-years of effort, excluding the original base application - mostly RPG III.

11. Procuro Systems

Procuro Systems, in contrast, has developed what is widely recognised as the leading short term insurance broker administration system in South Africa, parts having been developed on System 36 and System 38 since the mid 1980's. The system was developed using DB2 for i and RPG III. Old CPF Rel 2.0 "database thinking" is evident (no keys of PF). Investment in intellectual capital is approximately 40 man-years.

12.

In both cases, the functionality offered by the applications was considered to provide both organisations with significant competitive advantage, with substantial intellectual capital invested in these applications. The following facts were identified:

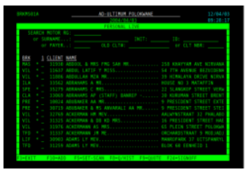
13. Massive value and investment

Both companies, after critically considering the functionality offered, and distinguishing between operational and functional deficiencies, recognised that there remained much value within their heritage applications. They also realized that their competitive advantage was essentially "hidden" within their heritage application. If this could be recovered and encapsulated, significant additional ROI was possible.

14. Monolithic Code

Due to the age of the applications, very little modernization had taken place over the years, with almost no re-architecting or separation of function. This means that data relationships and validations were still enforced within the RPG application logic, with the "database" essentially a flat file system. Code was predominantly System 38 style DDS file definitions (DSPF, PRTF, PF and LF) and RPG III.

15. Poor screen standards



Due to the age of the system and the fact that they pre-dated CUA, screen standards were poor and a lot of inconsistencies and gradual standards existed, which made the use of any "screen scraper" technology cumbersome. Additionally function key usage was in line with old System 38 thinking, which is problematic at best.

16. Maintenance backlog

What is however most interesting to note, is how every single installation we have been exposed to, struggles with the maintenance burden. Most surprising though is that few of the installations have made the connection between the maintenance burden and the cause for this problem: their monolithic code and that they never did any modernization, as “our system is working fine”. The maintenance burden in our experience is the biggest cause for end user frustrations and for looking at replacement solutions.

17. Old programming tools

If we take a hard look at ourselves, entrenched, as it were, in our comfort zones, it is not difficult to see that few of us invested the time and effort to improve our coding practices, learn new skills and adopt new technologies and tools, so WE have become the biggest risk and impediment to our system!
How many of us have truly become experts with RDP? It is a tool that can improve our productivity and accuracy by orders of magnitude, yet most developers still trundle along with PDM and ADTS??!! This is Ludicrous! Not to mention how few programmers have made ILE their own, developing small re-usable pieces of code.
Why the immense resistance to SQL? Why have the bulk of programmers not stayed current with latest programming models? Why no separation into multi-tier architecture (MVC)?

Please understand that this is a massive accusation against us all. We have the best commercial online transaction processing platform in the world, yet WE are the cause of most of the platform’s “ills”.

18. Anomalies

Additionally, due to the sheer age of the applications and regardless of the management practices, a LOT of anomalies have been introduced into our systems over time. Some of this was caused by “emergency” maintenance, but a LOT of it is simply caused by neglect – inconsistent validation and relationship rules in our code.

19. Special considerations

In Cape Gate’s case the UI was the least of their concerns, as they run a typical heads down, blue collar data capture production environment. In that kind of environment, limited function character based workstations remain the best solution.

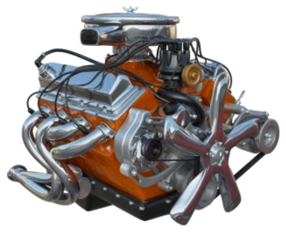
In Procuro’s case the UI was the most important immediate requirement, as they could no longer compete with other “flashy” graphics based applications, even though the functionality offered was leagues ahead of the competitive offerings.

20. Impact?

Due to the age of the systems and the monolithic nature of the code, the maintenance burden became absolutely excessive. The users in both these instances were happy with the functionality offered by the systems, yet were hugely frustrated with the lack of AGILITY and the delays involved when introducing changes. Additionally neither installation could afford “double maintenance” with any new or interim solution. In short, both of these installations foresaw an accident on its way.

21. Accident on its way
If they did not PRO-ACTIVELY start looking at ways to satisfy their internal (and external) customers, they were bound to lose their applications and most likely their IBM i completely.

22. Mitigation? Solutions investigated.



So, how did the management teams respond to this problem?

23. Full system audit and replacements

In both cases management critically assessed the functionality delivered by their applications, recognised the massive investment into their applications and documented the operational constraints that inhibited/shackled their AGILE response to changes in the business environment. As a result, they understood exactly what functional fit their applications delivered. In both cases their applications delivered higher than a 95% functional fit to their requirements, but more importantly, the management recognised that their applications provided them with competitive advantage. Armed with their functional requirements, they then approached vendors to assess what was available in the market, what it would cost and what level of functionality could be delivered.

24. Problems encountered with these solutions

To their shock, CG realised that the best functional fit they could get from the latest and greatest ERP applications available, was in the mid 60% range! In their words “we could get all the bells and whistles, in the latest “gee whiz” presentation layer, but essentially ALL their competitive advantage would be lost. Additionally, if they had to add the lost functionality, they would lose a number of years developing that functionality, whilst essentially running two systems in parallel (and ending up with the same result). The combined cost was simply more than they were prepared to consider. Another realization was the fact that these replacements would have required the replacement of IBM i and re-skilling of their staff. Replacement simply meant: throw everything you have in the trash can.

Due to the significant value proposition, with which they were intimately familiar, having run the platform and its predecessors since 1986/7, combined with the reliability, this was something for which they had little appetite.
They then extended their study to review what, if anything, could be done to modernize their application portfolio and launched a pilot project to review the results. The message from top management was such that the value in their current application made significant commercial sense to leverage this if at all possible. The initial results with IBM i based modernization tools was not very positive, for a variety of reasons. Some of these included inter alia, the nature of modernization projects at that stage (“Big Bang”), the perceived risk, the disruption, availability of skills, the fact that many of the solutions was what can be best described as “band-aid” solutions. One of the tools was in fact purchased, but their experience with it, especially its use of “surrogates” as an interim step, dissuaded them from continuing with the tool. Additionally their modernization project had to be worked into their already demanding maintenance schedule, due to several operational realities.

Procuro realised very quickly that with their system being a custom bespoke system they had to consider a complete redevelopment with another mainstream development environment. They did consider and cost a complete .Net, JAVA or PHP based redevelopment (their MD in fact did his MBA thesis on his experiences). They very quickly worked out that in the current economic climate they simply could not afford this disruption. They were essentially treading water, trying to remain competitive whilst redeveloping their system. They then started a pilot project to study the options available to them on the UI side, which was their biggest short term inhibiting factor. They quickly realised that this was a double edged sword, as fundamentally they did not correct the actual cause of their problems - namely AGILITY and secondly, by implementing a UI-only solution, which would have invariable caused double maintenance, they would in the longer term cause even bigger problems.

25. The Solution? Strategic Modernization

Based on their extensive research of available solutions and intimately understanding that they were confronted with a strategic choice, not a tactical one, they engaged with us on our approach and modernization roadmap which we term: Strategic Modernization.
It is driven by the following fundamental considerations:

26. Dealing with the problem

Firstly it is very important for a company to decide how much value remains in its heritage applications and the nature of the operational constraints caused by the applications.
Secondly there is a need for the company to prioritise addressing these constraints and to attach a monetary value of the anticipated value that will be gained by removing them.

Thirdly, a CRUCIAL consideration at this stage is to determine what constraints will be tactically met by "band-aid" solutions and how much will be invested in leveraging the fundamental value of their heritage application.
It is in our opinion quite realistic to expect to extend the life of the application, whilst remaining competitive, by another 10 to 20 years, if the competitive advantage is leveraged and the operational constraints removed. How is this achieved??

27. Legacy systems

In the past, we used to develop our applications in such a way that all applications logic, including all database (relationships and validations) logic, user interaction and discrete business logic were all encapsulated in large monolithic programs. It was for instance not strange for mainline transaction processing programs to consist of tens of thousands lines of code - a programmer's nightmare.

View

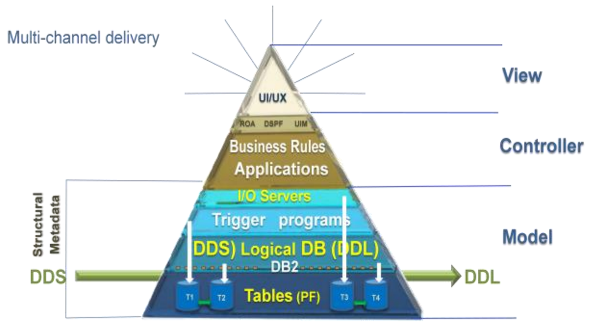
Controller

Model

By simply driving database relationships and validations out of the code into the database engine, the lines of code can easily be reduced on average by 80% (rule of thumb). The benefits of this are dramatic – see various resources at the end of this presentation and ask for case studies.

In order to do this, we need a solid foundation as basis for any lasting modernization drive. In our opinion, this starts with migrating/upgrading as much as possible of your database from the old DDS definitions to DLL, which opens a considerable amount of value and benefit. Lets study this closer:

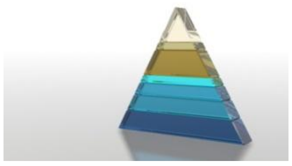
28. Modernize. How?



It is our contention that the only lasting modernization possible HAS to start at the fundamental database definition level. Any other adjustments or manoeuvres are tactical moves at best. It will not in the long term remove the fundamental barriers to a permanent solution. The challenge is how to achieve this with the minimum disruption to your users, the lowest risk, in order to have a SOLID, long term foundation to now start extracting maximum benefit from your heritage application. It is imperative to get your database definitions into DDL, with the bulk of your relationships and validations moved out of your application logic into the database engine. This will provide you with the foundation to start leveraging the incredible capabilities of SQL and the SQE engine. You absolutely want DB2 to do all the “dirty” work for you, allowing you to focus on delivering innovative business solutions and logic.

29. Transform DDS to DDL

Therefore your entire database definition needs to migrate to the SQL “personality” on DB2, as opposed to the old, antiquated RLA (record level access or native IO) we used to implement. A CRUCIAL consideration throughout and with every phase of your modernization project, is what the value (ROI) will be to the business. Our entire approach is designed to achieve EXACTLY this.



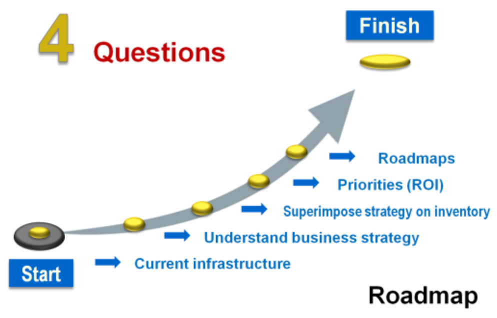
Non-disruptive; low-risk; gradual; full spectrum modernization based on value - ROI

30. Important questions
Is this realistic, feasible and actually sensible? And how far should you go in your quest to reclaim the value from your heritage? What questions should you ask? What considerations should guide you? Additionally bear in mind that you should ALWAYS aim to deliver positive ROI within 3 months of completing a project step. If you cannot guarantee that, think twice…

31. Blueprint recovered

As highlighted earlier on, once you have performed the initial migration/upgrade to the SQL personality of DB2, by moving from DDS to DDL, this is then used as a foundation to gradually, in a non-disruptive, business-as-usual fashion recover the essence of your system where there remains significant value.
We are not advocating here that you modernize your entire system, but rather that you be guided by the potential value to the business and improvement in the operational efficiencies. This should be achieved doing your standard maintenance, which usually implies that the most maintained parts of your system will be the first to be modernized. In our experience you will end up modernizing less than half of the lines of code. Be guided by the 80/20 rule.

32. Roadmap - 4 Questions:
Where are we? Where do we want to be? How will we get there with least impact/resistance & best ROI? Within what time frame?

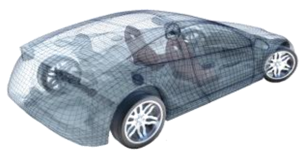


The way in which we develop the modernization blueprint jointly with customers, is to take stock of exactly where we find ourselves, to then define what their ultimate objective will be, within what timeframe they want to achieve this. Discrete projects are then defined to achieve this with minimum disruption, and KEY: maximum participation.
People HATE change so you need their ‘buy-in’. Our entire process and approach is to achieve maximum buy-in and participation from all stake holders. How does such a project usually evolve?

33. DDS to DDL

As stated categorically from the start, we believe that our reluctance to move to DDL and use DB2 the way it was designed, has caused us irreparable harm.
In our opinion, if you want to extend the life of your heritage applications and reclaim the value within those applications, you have absolutely no alternative but to move your database definitions to DDL as a matter of absolute priority.
There is DRAMATIC value and benefit – see the list of resources at the end of this. Study at least the document titled: "DDS and SQL - The Winning Combination for DB2 for i - September 2012 Update.pdf"

34. ROA and UI tool of choice

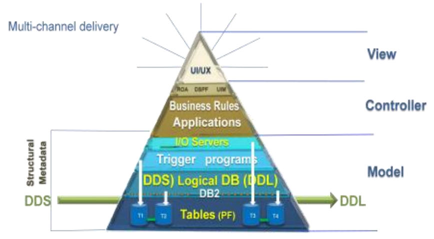


Our character-based UI has certainly added to the "legacy" perceptions. Quite often by addressing this perception or constraints, you can buy yourself time whilst implementing more lasting strategies. Another recent development which excites us is the announcement of Rational Open Access for RPG or ROA. This development, combined with the Open Access Metadata Open Standard and the UI tool providers (or your own handlers), completely removes the shackles on the 5250 data stream and your IBM i applications. Several of the UI tool providers have embraced this very important development and can certainly assist you in changing the "legacy" perceptions of your heritage application.

35. Design recovery at own pace

Once you have removed the most obvious constraints, you then have the opportunity to zoom in on those business functions where you will be able to extract maximum value for your company, in order to re-factor those functions. Only aim to modernize those functions which generate the bulk of your transactions. We always approach this from the 80/20 rule perspective, where experience indicates that 80% of your transactions are generated by 20% of your application. In an ERP environment that is usually order entry.

36. Modernize. How?



Lets study again what our optimum application architecture should look like and which elements are non-negotiable going forward. Implementing this, we can truly reclaim our rich heritage, whilst removing those building blocks which have caused a LOT of pain for IBM i developers. It is significant that if modernized properly, you may end up maintaining as little as 10% of the lines of code you had to maintain originally in your legacy application. Question: If you only have to maintain 10% of your application, what are you going to do with your extra time??? INNOVATE!!!

37. The surrogate

Please understand that IBM still becomes extremely uncomfortable when we talk about the "sensitive" subject of surrogates versus native modernization. The essence of the suggested surrogate solution as proposed by IBM since the early to middle 2000's, was their perspective that installations needed to start using the SQL engine or personality (WHICH WE AGREE WITH 100%) whilst insulating (masquerading) the change to the heritage application, by introducing a "surrogate", in order to prevent massive recompilation of your legacy application. The reason for their perspective is that they wanted to provide the benefits of SQL to new application developers and the newer development environments, where your typical .Net, JAVA or similar language programmers are very familiar with processing the database directly with imbedded SQL.

We however contend that you will unlock more value by leveraging the database of your heritage application, whilst providing a solid foundation for future growth, where you can then choose the best tool for the job. In terms of heavy IO and intensive commercial processing, RPG IV will outperform ANY of the other languages hands down.

Additionally, you then provide a single framework to facilitate your programmers working with result sets, as opposed to thinking one dimensional IO. Suddenly the same result set can be consumed by RPG, JAVA, PHP or whatever language is best suited for the job at hand. Think about the consistency.

The questions you want to ask yourself again are: How much value remains in your heritage application? How much more do you want to leverage that investment? And then, how do you want your heritage application to access DB2 – natively or via a surrogate?

Before you answer that, consider that approximately 80% of the lines of code (all lines of code implementing validations and enforcing data relationships) currently in your heritage application, will eventually and over time end up in the database engine.
Then consider where you want to go with this application. We believe a native migration is a natural long-term solution. So, do you want the surrogate masquerading to your heritage application, or do you want the full might of the DB2 athlete at the disposal of your heritage?

38. Clean-up

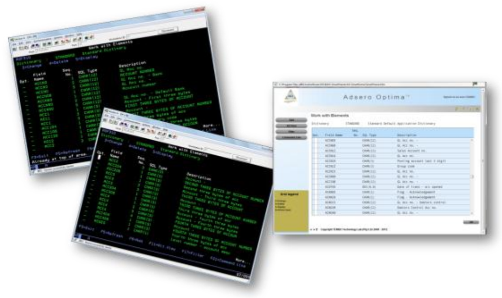
Due to the age of our heritage systems, a LOT of undesirable consequences have established themselves in our databases, our metadata and our structural metadata. By gradually cleaning this up, again in line with our entire non-disruptive, low risk, gradual approach, your database and system will become a LOT more AGILE and the integrity will improve dramatically. You will be amazed by the sheer amount of duplication that has occurred over time at a metadata and structural metadata level.

39 Open Access Metadata Open Standard

Another initiative which we believe holds immense promise for the entire IBM i industry, is the OA Metadata Open Standard, where the objective is to enrich our metadata with attributes which we seldom required in the DDS/DSPF/PRTF paradigm, but which can add immense value and AGILITY to any new systems or interaction mechanisms (read any UI). Think here of attributes such as standard fonts, colours, icons, urls, other behaviour. The potential is limitless.

<OA>
Open Access
Metadata
Open Standard

40. Before and After



This is a small example of the kinds of results you will quickly see once you start sanitising your metadata and structural metadata.

41. Gradual clean-up; parallel migration

One of the key considerations of our entire modernization philosophy and roadmap, is to gradually, business-as-usual modernize your system as you perform routine maintenance. The benefit is immense and you implement the very desirable "continuous improvement" philosophy. You will stand amazed at the results.

42. Evolution, not revolution

It is all about a gradual, non-disruptive, natural improvement of your heritage, in a low-risk, non-threatening way. You can drive the vehicle of your dreams, for much less effort than you anticipate. Don't our users, our platform and our applications deserve this?

43. No lock-in

We do believe that you should never exchange one dependency for another - quite often more proprietary dependency. Take ownership of your applications and the destiny of your systems.

44. Reaction to date

45. Excitement – modernized system

46. UI/UX – new look

Think multi-channel, cloud. Do not design for a single UI or delivery platform.

47. Agility

Implementing best practices, small modular components based on multi-tier architecture (MVC) will provide you with an application that will be AGILE beyond your wildest expectations. Additionally your testing, change and configuration management will become a breeze…

48. Modern programming tools

If there is one message we want to stress it is this: please invest, become absolutely familiar with RDP and sharpen your ILE and RPG IV skills. We have become our own worst enemies. With the combination of DDL, RPG IV, ILE and RDP you will wipe the floor with the competition and in the process have immense fun.

49. Bright future

We are quite excited with the potential PureSystems holds for IBM i based applications. Start reading up on this!

50. Future investment into IBM i-based applications assured

51 IBMi unshackled

52 Modern UI/UX

53. RPG and DB2 – a market leader in high volume commercial OLTP environment

54 Companies re-dedicating themselves to RPG and IBMi, as a result of ROA

55. Small, incremental steps, fast ROI.

56. Where do we stop

As stated several times, only modernize those elements where you can prove ROI very quickly, or break the objective down into more granular elements. Do not step into the trap of modernizing for the sake of modernizing. There MUST be substantial business benefit.

57. As IBM i users we have a bright future!

58. Where to find more information

IBM White Paper: DDS and SQL - The Winning Combination for DB2 for i - September 2012 Update.pdf
IBM i Strategy and Roadmap - July 2012 Update.pdf
Redbook: SG24-6393-00 (Modernizing IBM eServer iSeries Application Data Access - A Roadmap Cornerstone).pdf
Redbook: SG24-6503-00 Stored Procedures + Triggers and UDF on DB2 UDB for IBM i.pdf
Redbook: SG24-6671-00 IBM i Application Modernisation - Building a new interface to Legacy Applications.pdf
Redbook: SG24-4249-03 Advanced Database functions and administration on DB2 UDB for iSeries.pdf
IBM Presentation: Database Modernization - Object Creation and Access.pdf
Article: From DDS to DDL - How to fully exploit DB2 for i - Mike Cain - IBM Rochester.pdf
IBM presentation: IBM DB2 for i Indexing Methods and Strategies.pd
IBM publication: IBM i - Rational Open Access for RPG.pdf
IBM article: Modernising Database Access - The Madness behind the Methods by Dan Cruikshank.pd

All these available on www.adsero-optima.com/resources

59. 7 key points to take home:

- Huge value in heritage (legacy) applications
- Modernizing them feasible and realistic
- Do this gradually, business as usual, based on ROI
- Upgrade your skills: RDP(i), ILE, RPG IV Free, XML, HTML
- Move to native SQL engine (DDL & DML) as PRIORITY
- Implement multi-tier architecture
- Exploit ROA & OA Metadata Open standard
- PureSystems

60. How to contact me: Marinus Van Sandwyk

mbogo@tembotechlab.com
www.linkedin.com/in/mbogo