March 2019

Delivering Rapid Change in a Mainframe Environment:

Application Discovery and Intelligence

Sponsored by IBM

Nigel Willie
DevOps Practitioner

Introduction:

The role of technology within industries has changed and will continue to change. Previously technology was often viewed as an enterprise cost. While cost is still a key factor in all enterprises this is now coupled with a greater focus on the business value of digital capability. In many cases the user experience provided by technology is a business differentiator.

Digital innovation, transformation and automation applications are now standard across most major industries.

In my experience legacy applications are often seen as a challenge in the enterprise technology landscape. This technical debt becomes a problem when organisations let their software become non-current because of the challenge of prioritising evergreening activity when digital delivery is seen as a higher priority.

Problem statement:

Many industries have seen disruptive competitors enter the market. These companies tend to be small, innovative and agile delivering new capabilities rapidly. They employ modern technology and capabilities to enable them to innovate rapidly and deliver change at pace.

Established organisations tend to own large established IT systems, which while in many cases are extremely robust, scalable and cost effective may not be best suited to the rate of change now required.

The introduction of new competitors has also coincided with a period where the world economy is not significantly expanding and as a result cost pressures dominate in many industries. This makes the pressure to modernize in a cost-efficient manner more acute.

Scope:

Having worked with IBM Z for a number of years I have experience of the challenges faced in modernizing capabilities working with an established technology platform.

In this paper I will concentrate on some of the challenges I have experienced and identify potential paths or capabilities which would mitigate some of the issues and enable an enterprise to position itself better to deliver change to market more rapidly.

The paper does not feature many references to specific technology but is aimed at discussing capabilities that would assist in meeting business challenges. I have deliberately focused on capabilities above specific platforms as I believe this demonstrates that a specific technology is not the constraint but rather the culture and practices.

The term ‘established technology’ is used rather than ‘legacy technology’. The term legacy tends to focus on the technology as the problem rather than the outcome. Many ‘legacy’ technologies support current profits.

I shall particularly focus on the challenges around application intelligence and discovery in this whitepaper.

Understanding the challenge:

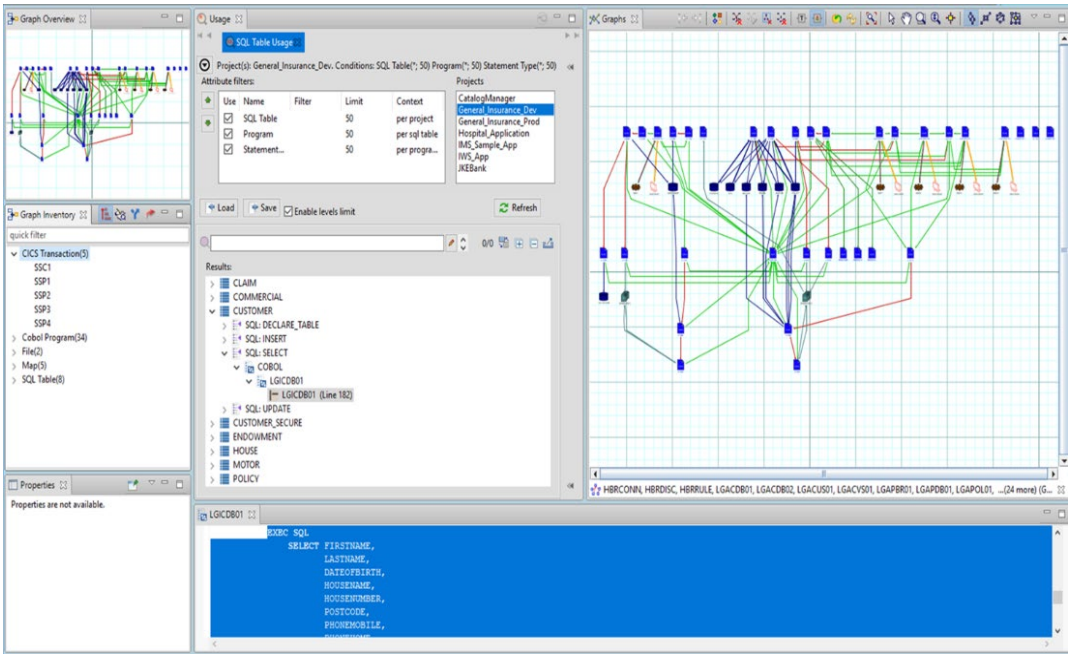
Regardless of technology, the key challenge facing most companies when faced with driving greater agility into their existing technology is technical debt. Whether this be large monolithic solutions, poorly documented code, loss of corporate intellectual property as the individuals who created the product are no longer present, or a lack of test assets. The fundamental issue is technical debt.

I shall start by detailing some specific problems and potential approaches to mitigate these.

1. *Quality of Documentation:*
   Many applications are historic, have old or non-existent documentation. Often the documentation is in text format and verbose rather than technical. The same document has not been updated when the asset was which leads to inaccuracies and introduces risk.

   Additionally, if we wish to introduce greater agility it is critical that documentation is not only accurate, current and succinct but that we continue to keep it in this state.

   To mitigate - a move from heavily text-based documentation to automated diagrammatic representation of data and logic flows makes it easier for technologists to make rapid change with reduced risk. If this information can be refreshed in real time from the asset itself, we provide a step change in capability.



Analysis example showing application inventory, SQL usage and interdependencies

2. *Understanding Dependencies:*

To make change you need to understand dependencies. Ideally in time we architect away many of these dependencies. We should recognize that this is an end state rather than the current reality for most companies. The more rapid the change, the more you need to understand who or what could be impacted.

To mitigate – We need to understand both dependent applications and products but also dependencies at a database level. If I increase the length of the account number field where else in my estate is this read, altered, stored etc.

3. *Creating APIs to decouple architecture:*

To reduce dependencies and move toward more easily changed micro services we need to abstract our technical assets. To do this requires APIs. These need to be defined at field level.

To mitigate – It is critical that we understand that all fields are consistently defined across the estate (the same size and attributes), that we validate who has a dependency on an API to ensure we can contact them should new features or a higher version be released and to ensure nobody is circumventing the API and calling a database field directly. To be able to achieve this with a level of confidence in an ever-changing landscape we need any solution to provide all this information on demand across current and any 'in development' assets.

4. *Understanding the current picture in a changing world:*

As change becomes more rapid, then the ability to document, record and refresh your current state and dependencies becomes more critical. Traditional means of recording your estate using word documents and technologists as authors become obsolete.

To mitigate – Whatever solution you choose to record your current estate must be capable of being refreshed in real time with minimal effort. There are functionally excellent solutions available that unfortunately impose significant manual effort to provide an updated view. This results in a time lag in information being provided and a consequent reduction in confidence and increase in risk.

5. *Understand what is critical:*

Any change introduces risk. Most enterprise development shops are well established and can have significant complexity. In my experience a significant minority of production outages are caused by unintended previously unrecognized, dependencies in applications which were not part of the change itself. To reduce this risk change is often delayed as large-scale integration testing takes place with many applications and their associated environments involved.

To mitigate – Be aware which applications are most critical to our business. By understanding technical dependencies, we can validate which, if any, of our most critical applications have dependencies on our change. The ability to do this enables us to rate the level of risk and take a more informed approach to release.

6. *Understand what is not required:*

Simplification is a key to reduced costs and increased speed of delivery. One approach to simplification is to ensure that dead code is removed. By demising unused code, we increase performance but also reduce the effort of maintaining the overall code base. Historically companies have been very poor at identifying unused code and this technical debt adds cost and risk to their operation.

To mitigate – Being able to accurately identify which code is not used and acting on this information is a major enabler to efficiency. The ability to not only recognize programs which are no longer used but also code within active programs which is dead is a key value add.

I have seen migration initiatives expend significant effort understanding code which has ultimately proven to be dead. The ability to identify this fact quickly and accurately would have saved significant resource.

7. *Move from projects to products:*

Agile and DevOps thrives on long lived product teams, building and supporting a product that has a business owner. It is likely that current solutions have been built using a project approach and you may need to re-architect in places to ensure your solutions are based on product capabilities.

To mitigate - Understanding what attributes, the business product has is the first step, you then need to identify these within your technical estate to ensure you can define the technical product the team are accountable for.

8. *Understand the environment:*

The adoption of a DevOps culture is a significant initiative in the industry. The key outcome of this culture is delivering incremental change to production rapidly. The key enabler is the ability of development and infrastructure teams to collaborate and work more closely together in a move toward multi-skilled individuals managing the full technical dependencies of change. Enterprise shops have traditionally siloed these roles.

Speed of change relies heavily on the fact that you have confidence in the fact your run time environments at all stages of the test cycle are the same and that production will be just another environment.

To mitigate – Ideally to enable this capability we need to understand and have an inventory of each of our environments. This enables us to recreate the intended environmental conditions at each stage of the change and increase our confidence in our testing being valid and efficient.

9. *Contextual understanding:*

So far, I have detailed technical challenges. Let's now look at understanding the business context of the asset the technician owned.

In the industry one of the weaknesses that can lead to poor quality decisions and solutions is the inability of the technologist to understand the business context within which their solution sits. The impact on others of change made in a silo often leads to decisions that would not be made if context was understood.

To mitigate – Above the technical value that understanding dependencies brings, the ability to understand who depends on your solution can drive positive behavioral change in developers. DevOps is appearing as a framework to gain agility tied to business value.

10. Maintain the asset:

The ability to deliver change rapidly is negatively impacted by technical debt. Equally delivering change rapidly introduces the risk of focusing on functionality and in doing so increasing the technical debt and introducing problems that will act as a brake on future change.

To mitigate – The industry uses long established metrics such as cyclometric complexity, Halstead complexity and maintainability metrics to enable technologists to measure trends in their product. The ability to utilize these same techniques in a mainframe context enables us to manage the risk of technical debt.

11. The demographic challenge:

IBM Z developers tend to have more experience than certain technologists. If you intend to invest in IBM Z, or even maintain it, you may face a demographic challenge. Although COBOL is just another programming language, new developers are also asked to learn critical Z subsystems (CICS, IMS, Db2). Most new technologists will be used to working in a world with eclipse technology, instant feedback on change and the ability to drill down into dependencies.

To mitigate - To attract and retain talent you will increasingly need to offer a similar experience to your technologist to that they experience on other platforms. The ability to offer more modern tooling is a significant aid to attracting and retaining younger talent into the enterprise space.

Options:

Faced with the challenges outlined above there are normally two directions that can be taken with established technology solutions:

- Demise and migrate
- Modernize the applications coupled with your processes and practices to compete.

Demise and migrate:

This approach can be most suitable if the cost and effort involved in modernizing existing solutions is deemed too expensive and / or too time consuming. Sometimes it makes more sense to start again rather than modernize.

A decision to demise and migrate should not be taken lightly. There are many examples of this approach delivering huge costs and delivering no, or little, value. It is not a simple task, indeed the more complex the solution being demised the more difficult the task of delivering the new solution. Whether the new solution is procured or developed, the effort required to fully understand the scope and integrate into the rest of the estate is often underestimated – and there are many examples of projects abandoned part way through because of significant overruns.

The potential advantages are, that provided the new solution is correctly architected, designed and built, the enterprise is likely to be in a better position to deliver change once migration is complete, assuming that the application is a match for the platform's qualities of service. It should however be remembered that if you build a solution that reflects a competitor's current strengths their solution will have advanced in the time it takes you to migrate.

Modernize your practices - *Embrace CI/CD:*

The challenges here are also significant.

The ways you work are going to have to significantly change. Moving to agile and iterative delivery is a cultural challenge in any organization.

Often the need to change is recognized more clearly outside the immediately impacted teams than within. IBM Z teams tend to be well established with processes and cadences which individuals find comfortable. While many of these processes will need to change it is critical that you recognize many will still add value in the new organization. It is very dangerous to ignore existing knowledge. The difficult balance is to help the teams to understand what needs to change and embrace this, while understanding what is valuable and needs to remain.

Abstract your IBM Z assets:

One pattern I have seen regularly exhibited is the architecture of an organization imposing a need on all applications to move at pace in order to deliver change. It is not unusual to see a multitude of various calls to Z products all of which effectively serve the same business capability.

In this scenario IBM Z can historically be seen as an inhibitor to change, as digital capabilities are built rapidly in isolation with no awareness of dependencies, leading to late change requests to enterprise teams, a view that they are the bottleneck and further desire to 'remove the legacy'.

A well architected estate can remove some of these pressures and drive efficiencies. By creating API's in front of IBM Z assets we abstract these from change and avoid them becoming a dependency on every initiative.

In a large organization it is likely that each of the above options above are valid for some applications. Regardless of the option chosen the challenges are similar and will need to be resolved.

In conclusion:

The rate of change has increased significantly in recent years and all enterprises need to react to ensure they remain competitive. All change introduces risk.

Regardless of your enterprise strategy the ability to understand the assets you own and to rapidly analyze impacts of any proposed change on these is critical to reduce risk and increase velocity.

There is nothing inherent in IBM Z as a technology that precludes you from adopting the agile approach to change prevalent in other technologies – indeed the technologies are now mature for DevOps on Z to be rapidly adopted.

One of the first steps to enable you to achieve this, and continue to do so safely, is the ability to understand what you own and the impacts on this of any changes you make. The capabilities and approaches detailed in this document can be used to add value to your enterprise. One leading example of such a platform is IBM Application Discovery and Delivery Intelligence (ADDI) that has the ability to work across most common languages and databases and provides a solid foundation for the Digital Transformation initiatives that are a current priority for most organisations.

About the author:

With over 20 years' experience working in IT for one of the world's largest financial institutions I have experience of managing and delivering global transformation programmes. My technical background was initially on the mainframe platform.

My most recent role was to deliver cross platform DevOps automation capabilities to the enterprise for HSBC. This included the delivery of mainframe development automation capabilities.

I am a great believer that introducing capabilities such as continuous integration and continuous delivery comprise a cultural and architectural change as much as a technical change. I further contend that this change can be embedded across technical platforms. The challenge may be greater in some areas but there is no logical reason why the mainframe platform cannot support and contribute to rapid delivery of iterative change at enterprise level.

78024478GBEN-00