PLURALIT

INCLUSIVE TECHNOLOGY

MANAGING ENTERPRISE LEGACY SYSTEMS

In a “Devops world” the whole team is only as effective as the weakest link. Unless well managed this is often the enterprise legacy.

White Paper | April 2023 (R1)

SUMMARY

BACKGROUND

Managing legacy systems creates numerous challenges. Systems running on old technology become unstable, are no longer supported by vendors or run on technology where development and maintenance skills are hard to find.

These challenges augmented during recent years by a vicious cycle that led to a shortage of critical skills for older systems. As seasoned specialists in legacy retire, younger system architects and software engineers prefer the latest technologies and avoid legacy systems, perceiving them as lower-grade, CV-limiting work. And those few that can be interested will have no experience in older systems.

The business risks depending on legacy tend to be disproportionately high, as are the limitations older systems impose to market agility. By definition, legacy systems will have been in operation for years and hence support key enterprise processes the business has come to rely on. As legacy systems become unstable and/or difficult to change, so will such key processes. Furthermore, as the company seeks market agility -faster rollout of more targeted products in an efficient manner- in a competitive landscape, its legacy will inevitably become a blocker for such goals in one or more of time, cost and quality of output.

So the temptation to leave Legacy systems well alone will only put off a snowballing problem. Eventually, an essential business process change will be needed, a desired market move will become critical, or a piece of old infrastructure will totally fail. By then, the urgently needed legacy change must be handled with extreme care or it could create further instability and blockers. In this situation, it is almost impossible to perform a major strategic upgrade to introduce new features without replacing the whole system. Doing this requires facing the challenge of reverse-engineering features built over many years, with the original software engineers long gone.

Most companies have recently identified an urgent need to leverage the latest AI technologies to get further value from their data. Cleaning up data and making it more accessible are therefore urgent challenges, which again are hindered by data residing in legacy.

It is therefore essential to keep on top of legacy systems. Diverting valuable resources to apparently less strategic maintenance of old systems will always be in competition with other business imperatives. It is essential to gain support from a company’s senior leadership right up to the CEO to manage budget tradeoffsand also clear the way to close down some legacy business processes which are otherwise expensive to support.

SUPPORTING LEGACY SYSTEMS

There are a number of recommended practices to support enterprise systems -legacy or otherwise- that every engineering team should embed in their way of working, to be ready to step up when an issue arises or new business features desired:

1. Maintain an accurate inventory of every enterprise application they are responsible for, linking to documentation covering key support steps – for example how to redeploy following a change. This should also record the versions of all operating systems and hardware platforms with particular focus placed on those out of vendor support.

2. Ideally, every application should have two members of the engineering team with a working knowledge of the code and some experience of making changes. The team should be constantly evolving and adapting towards this ideal.

3. It is important to avoid a large backlog of minor bugs which over time create an unstable environment.

SUPPORTING LEGACY SYSTEMS

For legacy systems, there are additional steps that will derisk their support and get them ready for stable operation, adding new desired features quickly and securely, and migrating to the cloud:

4. Engineering teams can mitigate structural legacy risks by engaging a “guild” of senior legacy experts with a list of proactive fixes and business readiness requirements for their older systems. These experts are typically senior system architects and software engineers with operational experience in older platforms and project experience in legacy data migration and readiness for AI data ingestion, cloud deployment of replacement systems. By having access to “legacy skills as a service” one can benefit from cost-effective base engagement plus timely access to multiple legacy expertise for peaks of demand - for instance a priority project or disaster recovery following an operational emergency.

5. The IT culture and organisation should recognise the value of legacy skills. In a “Devops” world, where engineering teams take full accountability for the systems they run, the whole team is only as effective as the weakest link. A great way of getting a team to value legacy and take accountability for it is to create a support rota where the role of legacy maintenance is periodically rotated around the team members.

6. In areas where instability is a key problem, or there are significant operational issues such as frequent spikes in demand, it often makes sense to establish a full-time role for a Site Reliability Engineer. This is usually a very experienced engineer who takes ownership of issues across a range of systems, manages expert contractors across multiple domains (including the legacy “guild” mentioned earlier) and coaches junior engineers. This role is obviously important for all mission critical platforms and frequently takes responsibility for creating automated deployment and continuous integration processes. The ability to safely make small incremental changes to a system - in particular legacy ones- is a vital part of any “Resilience Strategy”.

CHALLENGES TO MIGRATING LEGACY SYSTEMS

Often an engineering team will realise that their only way forward is to migrate their legacy system onto an entirely new platform. This may simply be to address stability or to support new business features beyond the scope of the legacy platform.

In most cases the preferred migration route would be to deploy the new system in the cloud to take advantage of the flexibility provided by cloud infrastructure. In some cases time pressures make it expedient to “lift and shift” a set of modules making minimal code changes but deploying to a stable and secure cloud infrastructure. Whilst this controls risks it may be very costly in cloud consumption since on-prem applications are not architected nor optimised to manage resource use efficiently.

In cases where future requirements are likely to get more sophisticated or traffic and/or volumes are set to increase, it makes more sense to put effort into re-engineering the system base so that it can take full advantage of the latest cloud tools and technologies and optimise cloud consumption.

In any migration it is vital to focus on the main business driver. If on-prem server and network hardware are failing and the most important issue is to rapidly migrate onto stable hardware, an on-prem solution could still be considered, perhaps as an intermediary step to a cloud migration in the end.

Migration projects face very specific challenges:

01

The old system must continue to operate while the new system is being developed. This may be fine unless the business can't wait for a new feature or the migration project experiences delays. This means the new feature must then be developed twice, for both old and new systems in parallel, and even some of the foundational design for the new system may need changes. This results in further delays and more parallel development of new requirements. This is one of the main reasons projects that follow a "big-bang" approach (where a new system is cutover in one go) often run into problems.

02

It is very hard to reverse-engineering software where the detailed requirements have been lost in the mists of time. Teams may often mistakenly second-guess the original developers, forgetting that business requirements may not have arrived in logical sequence. As a result, what appears very suboptimal code may have been a feat of ingenuity to adapt an existing routine to do something completely different. Assuming the original developers were "incompetent" leads to false assumptions about what the system actually does. This might only become apparent quite late when the new system is regression tested by current business users.

03

Systems replacing legacy have to achieve normal operations immediately. This is rather like joining a highway at 80mph from a standing start, and unlike the scenario for a brand new system where there is often a pilot launch and then a gradual ramp of users.

04

The most complex element of a legacy-to-cloud migration often relates to migrating data. Most legacy systems contain large numbers of data anomalies and over time code will have been written to work around these. Also, there often is a 'smart way' to phase data migration for lower cloud consumption during the transition, which may take several months or even years to complete without disrupting business operations; ignoring this may result in shocking consumption bills and trips to the CFO office. These types of issues can make migration extremely complex and it is vital not to underestimate the effort required to address data issues during a legacy migration project.

All of these challenges increase the risk of delay or even eventual project failure where the business gives up on the new system. It is very important to mitigate these risks from the start.

OUR RECOMMENDATION TO MIGRATE LEGACY: AN INCREMENTAL APPROACH (‘AGILE FOR LEGACY’)

The managed risk approach is often to break the project into small steps and carry out the migration incrementally. Teams may first consider this unattractive as it means right from the start they are accepting to support two systems and they may have to write additional code to break apart the original application. However, this ‘Agile for Legacy’ is the smart approach as we further outline below.

One key benefit is that at each stage there is an obvious fall back option. If a migration step doesn’t work it doesn’t invalidate the whole project. If the migration is from on prem servers to the cloud many operations teams will resist getting into a situation where they are supporting a mixed environment. An incremental rollout will create that type of complexity, but at the same time it means new processes for operating in the cloud can be honed in stages rather than having to work perfectly from day one.

The second benefit of an incremental migration approach is that it forces detailed analysis to determine exactly which elements of an existing platform are creating instability – and hence should be migrated first – or how the impact of developing a new feature can be reduced by isolating it from the overall system.

Following this ‘Agile for Legacy’ approach the business will experience key benefits much earlier and at lower risk. Although the overall project may take longer by design, it wil be significantly de-risked and more likely to meet design timelines. In addition, changes of course required by unexpected business requests can be responded to more effectively – this is essentially “Agile for Legacy.”

ARCHITECTURE

Another benefit of taking an incremental approach to migration is that, by properly assessing the architectural options, different approaches can be applied to different modules. Some can be stabilised by a simple lift to the cloud; others are recognised as needing a complete rewrite to achieve future business flexibility; and many will fall somewhere in between.

This also helps optimise expert engagements along the migration. If the business relies on external experts for different legacy systems (i.e. the "guild" mentioned earlier), it can plan increments around each system and hence engage less experts per increment, reducing both cost and risk (with fewer type of systems per increment).

One problem with legacy systems is that a stream of constant changes over time results in "bloated" modules with lots of inefficient, hard to maintain code. A key architectural trade-off is to balance two competing desires: to make minimal changes to de-risk the migration, and to build a system reflecting the latest technology. Breaking a system apart can deliver major benefits in terms of security by establishing trust and verify protocols between modules.

Creating stable API's between modules provides greater traceability of user journeys and provides a stronger basis for applying the latest approaches to securing systems.

CONCLUSION

Day to day management and migration of legacy enterprise systems is critical to the success of any business platform. It is frequently viewed as a chore and not given the focus it requires. Plus, expert resources are increasingly scarce.

Several recommended practices can reduce risks and surprises around legacy systems, such as: maintain an up-to-date systems inventory; prevent spiraling bug logs; evolve the on-prem architecture to stabilise legacy platforms; proactively prevent single points of legacy knowledge; strenghten legacy expertise in a flexible, cost-effective manner; and raise the legacy profile within the organisation with a Site Reliability leader, rotating accountability for legacy, and a culture that values "vintage" expertise.

For migrations to the cloud, taking an incremental approach ('Agile for Legacy') is key to de-risk the end-to-end project.

Across the way, both for supporting and migrating legacy, access to a "guild" of Legacy experts can help ease access to scarce skills and modulate demand level and domain expertise.