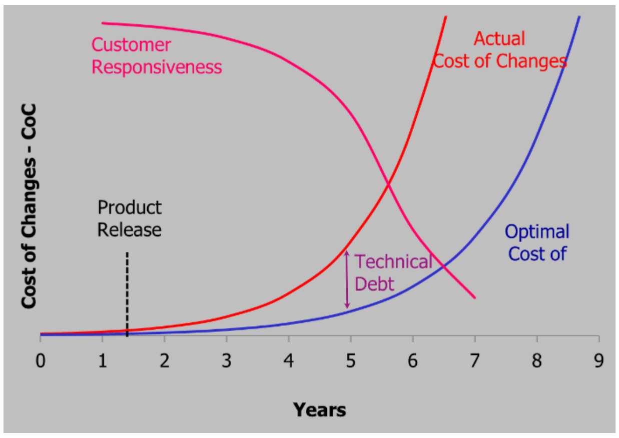
TAGAR
Computer Systems Ltd.

Modernizing into Business Centric Software Design

Advanced modern software architecture design, and software development design, are the key to robust and resilient software, yet easy to quickly adapt to changing requirements.

However, the majority of businesses base their operations on legacy software systems. Adding modern and required features to these systems gets, in time, harder longer and much more expensive (see following graph).



The farther to the right on the graph the bigger the pain to the business. In that situation the business can opt to select only one of the following strategies:

- Doing nothing - worsens the situation.
- Replacement by a new system (off the shelf as well newly developed) which incurs very high cost and very high risk.
- Modernizing the existing system.

However, unfortunately even after selection of the second or third path usually the organization ends up with the same faults that, in time, deteriorate the system to the same place at the far right of the graph. We propose a way to avoid this deficiency by designing business centric software.

We design software that does not require patches during its whole life cycle, thereby maintaining its architecture and preventing it from turning into spaghetti code over time.

Modernized software that lasts, rather than deteriorates over time, must be designed for that goal as business centric software is. Business centric software design suits both code replacement (but with considerably reduced risk) as well as gradual modernization of the code.

Business centric software design yields dramatic reduction of time to market. It also dramatically reduces development costs and mostly maintenance costs plus the huge savings incurred by maintaining the same software architecture intact with no need for re-writing the system every decade and a half (in average).

Implementation of software by Business Centric Design results in significant reduction number of code lines, thereby dramatically reducing the cost of both development and maintenance

It also allows for blazing fast response to requirements change request as well as ever-changing market challenges

What is Business Centric Design?

Software system is business centric only when it always achieves satisfactory response time to change requests and satisfies ALL the features the business needs. By ALL we mean those that were defined in the past, those which are known, those that shall be defined in the future and those which are unknown.

The methodology we use brings long term benefits because the building blocks we initially design cover all business domain's aspects.

Introduction of a new feature to the software does not change its architecture, it merely is an additional and new integration of the existing architecture building blocks.

After extracting the software's core use cases we identify areas of potential change in those core use cases. Each volatility is encapsulated into a software module that serves as a system building block, thereby encapsulating future change inside one module to prevent the change from affecting the whole system.

We do not design by the requirements list nor by documents based on the requirements list because requirements keep changing over time. These changes eventually erode and invalidate the requirements-based design. Alternatively, we design by meticulous business domain analysis since the business domain is very stable and rarely changes. Software architecture designed that way requires no changes over time.

Business Centric Design can be used for modernizing legacy software (Brown field) as well as for replacing that legacy system at once (Green field).

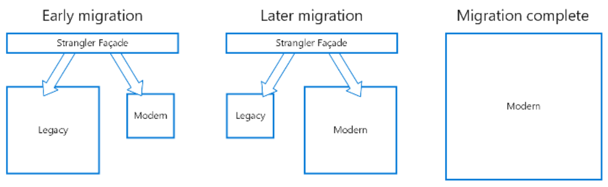
Modernizing legacy software can be a long-term process that gradually transforms the legacy software into a modern Business Centric system that supports all business domain's activities.

One option for modernizing as a process is by initially wrapping the legacy system with a lean shell of modern system. That shell is designed to modernize the legacy system via a step-by-step replacement of old code with new (see details below).

Modernizing legacy S/W as a process

We use the Strangler Pattern when modernizing legacy software should be conducted as process. That is a Façade that hides the legacy system and controls the software flow during the migration process.

Following is a figure of Microsoft's description of the process:



For more details: https://learn.microsoft.com/en-us/azure/architecture/patterns/strangler-fig

That way the migration becomes a well under control and much less risky process than applying a new system as a block.

Cost of utilizing that technique are also reduced when using the normal budget of maintenance and new features introduction to advance the migration by adding to the new system another vertical slice that executes the required feature rather than fixing or changing the legacy system. Very often clean slate writing is much shorter than fixing an existing old code.

This activity will normally require little or no extra budget for that implementation compared to applying it on the legacy system.

Business Centric Design's Benefits

Substantially improved responsiveness to everchanging market demands. Shorter development, testing and deployment time (days instead of weeks/months)

Substantial reduction of the number of lines of code without losing any required functionality

Guaranteed substantial SLA and cost reduction

Guaranteed development plan with accurate scheduling as well as guaranteed completion on time and within budget

Maintaining all required functionality without performance degradation

Extended system reliability also in distributed computing environments

Extended scalability not eroded by future changes

Higher customers' and users' satisfaction

Add almost unlimited features to legacy systems thereby solving technology gaps of older systems by modernizing them

Ability to implement gradual transition from legacy systems into modernly architected systems

Control of the pace of new features introduction to legacy systems and of their respective implementation schedule according to financial and other business constraints