

Golden Rules for Business Analysts

Colin S. Penn
IRM Training Pty Ltd ACN 007 219 589
Suite 209, 620 St Kilda Rd, Melbourne, Vic. 3004, Australia
Tel: +613 9533 2300

Abstract

Like all professions, business analysis has its golden rules – rules that are fundamental to the design of successful business systems. They might seem like common sense but it's surprising how often we forget them and get ourselves into hot water. Here's a short list of some of the more relevant ones...

The sooner you find a problem, the cheaper it is to fix

Get the specifications right

Recognise the total cost of a system

Don't design the solution before you've analysed the problem

What applies to small systems doesn't apply to large ones

Don't allocate conflicting roles in a project to the same person

Now let's have a look at what they mean, together with some real life examples witnessed by the author.

Background

You might feel that working with IT systems demands a constant refresh of skills and knowledge. Companies want efficiencies and reduced costs through new procedures such as Six Sigma, ITIL, COBIT. They must also accommodate new laws such as Sarbanes-Oxley plus IT governance and privacy legislation. People working directly with technology need vendor certification, be it Microsoft, Sun Microsystems, SAP, CISCO, Oracle...etc. In project management there are certification bodies – AIPM, PMI – plus industry accepted methodologies such as PRINCE2. Even the business analysis profession now has its own qualifications – CBAP and QBAP (see the glossary for what all these acronyms mean).

With business and technology changing at breathtaking speed, the choices with accreditation and certification can be bewildering. How we apply this knowledge however still relies on fundamental principles which don't change quite as fast, if at all.

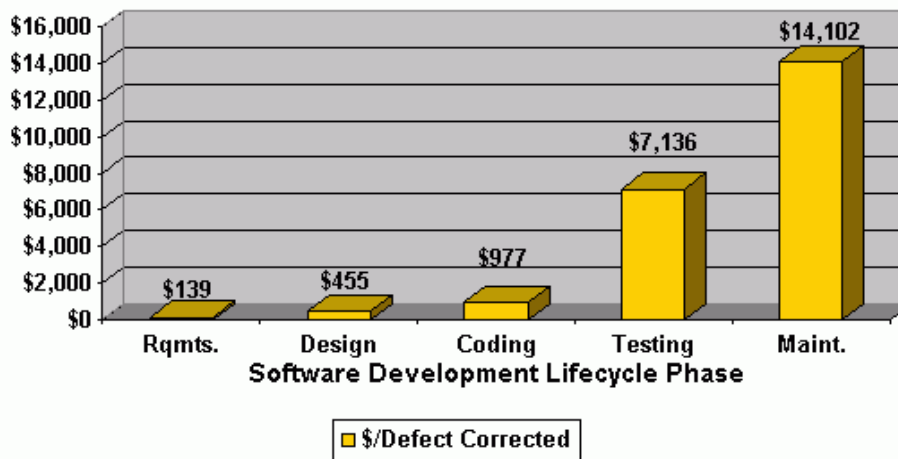
What are the characteristics of a fundamental principle – a golden rule? Usually it's a rule that can be applied whatever the business system or application, whatever the software or hardware platform. Consider the following and see how many apply in your environment.

1) The sooner you find a problem, the cheaper it is to fix

Barry Boehm¹ (noted software engineer and creator of the spiral development model) was one of the first people to consider how to estimate IT projects and where resources must be concentrated to achieve success. He showed that the cost of removing a fault in a system rose exponentially throughout the systems development life cycle. The following graph (courtesy IEEE Computer Society) illustrates the huge jump in costs the further along the development cycle that you go.

Costs of Correcting Defects

Source: B. Boehm and V. Basili, "Software Defect Reduction Top 10 List," *IEEE Computer*



Boehm's research showed that for every dollar spent correcting a fault in the requirements phase:

- \$3.27 would be spent correcting it in the design phase
- \$7.03 would be spent correcting it in the coding phase
- \$51.33 would be spent correcting it in the testing phase
- \$101.45 would be spent once the system went live

What does this mean for a business analyst? Let's suppose you've got a problem on your current project. You're in the requirements gathering phase and one of the requirements just doesn't make sense. To remedy this you need to organise an additional 2-hour workshop with stakeholders and managers to discuss and clarify the requirement. Let's say the cost to the company of this workshop is \$1,000 in people's time.

Now let's suppose you didn't spot the problem with this requirement until the design phase had started. Suddenly, the cost to your company of fixing it jumps to 3.27 times what it would have cost to fix it in the previous phase. (\$3,270 in round figures). If you didn't spot the problem until the coding phase the cost increases 7.03 times (\$7,030) and if you didn't fix the problem until testing it would be 51.33 times the initial cost. By the time your system is in production and under maintenance the cost of fixing your problem (which might originally only have cost you \$1,000) has now mushroomed to \$101,450 or 101.45 times the original cost.

¹ Barry W. Boehm, 1981. *Software Engineering Economics*.

With projects under ever increasing time pressures, it's often easy to overlook Boehm's Law "errors are most frequent during the requirements and design activities and are more expensive the later they are removed".

Case study - a city council developing a replacement payroll system believed that council staff (users) had comprehensive knowledge of all the business requirements. After all payroll was a highly procedural business process that staff had been performing on the previous system for several years. It was assumed that there was no need to analyse the current system and that resources could be better used elsewhere. However none of the current staff or IT team had participated in building the old system - they had no knowledge of how it was built, or the undocumented business rules it performed. When implemented, the new system could not correctly calculate employee benefits nor apply tax rules. The new system was eventually abandoned with all development costs written off.

2) Get the specs right

This golden rule is often called Glass's Law. You only need to see the title of Robert Glass's 1998 book *Software Runaways: Monumental Software Disasters* to understand the message "requirement deficiencies are the prime source of project failures".

To put some statistical perspective on this - in 1994 the Standish Group published ground-breaking research (the CHAOS report) showing that over 20% of all IT project failures were caused by incomplete or badly managed requirements.

Not a lot has changed today - research published in *IEEE Software* magazine (September 2008) claims that around 30% of software projects are cancelled or unsuccessful with one of the main reasons being badly managed requirements.

Additionally, the *Business Analysis Benchmark* - published in 2008 by IAG Consulting - surveyed over 100 companies with an average project size of US\$3m. The results found that over 40% of the total development budget was consumed by poor requirements.

Glass's Law further confirms the need for sufficient resources in the analysis phase. It ensures the greatest return on investment (resources). Thorough analysis catches more faults - removes them at the lowest cost - in the fastest time.

Case study - The CFO of a large corporation insisted upon a certain ERP software application being implemented. The IT department tried to insist on a selection process to consider a second ERP which would be a good fit to their infrastructure and their methods of doing business. However the company (having just come through difficult financial times) owed its survival to a strategy from the CFO so he won the political battle.

The ERP selected was new, untried and developed by a small vendor. Within a short time it was realised that the application's functions did not match the way the company operated. To make it fit, the company attempted to convert many of the functions in the application to work the way they wanted. The cost was high and they found they were subsequently unable to upgrade to the next release of the ERP because of all the changes they had made. After a short life span it had to be replaced.

3) Recognise the total cost of a system

Many people think only of the cost of developing or buying an application system. However, once it's in production the cost of running, maintaining and enhancing it over 5, 10 or more years, far exceeds this. Because these costs are spread over several years they are often ignored or glossed over. It's only when the accountants get involved and perform a Net Present Value (NPV) calculation of the total system cost over its total life span that the truth emerges.

Maintenance² constitutes the vast majority of the total cost of an application system throughout its complete working life.

Some maintenance costs are obvious, some less so. Some will depend on how the system is rolled out – client server, web services - others will vary depending on the use of existing infrastructure.

Consider the following list – are they development costs, maintenance costs or both? Sometimes the development system can be used for production, sometimes you will need a separate one. How will costs vary if a system has a 5 year lifespan – or 10 years? What if the number of users or customers grows by 5% a year – what about 20%?

- Development costs
- Development system
- User acceptance testing
- Production system hardware and software
- End user hardware and software
- Maintenance contracts (hardware & software)
- Rollout of production system
- Data centre (floor space, electricity, cooling)
- Rollout to end-user systems
- End-user training
- Network hardware & software
- Communications costs
- Backup and contingency planning
- Periodic disaster recovery/business continuity testing
- Mid-life hardware refresh and migration costs
- Software upgrades
- Personnel – maintenance team, help desk, operations
- Audit costs, insurance cover
- End of life decommissioning

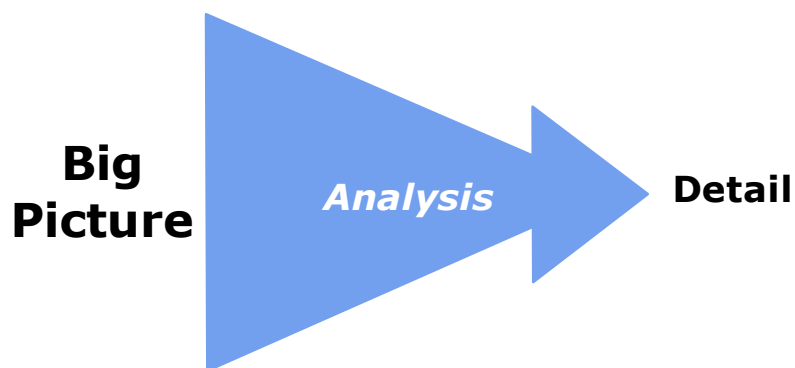
As a business analyst we may have little control or influence over these factors but we can make a significant contribution nevertheless. The business world is full of horror stories of undocumented systems – some of them at embarrassingly large companies. Do your bit by making sure the requirements documentation is as precise and unambiguous as it can be. In years to come, when your company wants to modify the business functionality of a system, it will be your requirements documentation they consult.

² *Maintenance is a generic term used to describe everything that happens once a system goes into production (goes live). It includes operational and enhancement costs as well as bug fixing.*

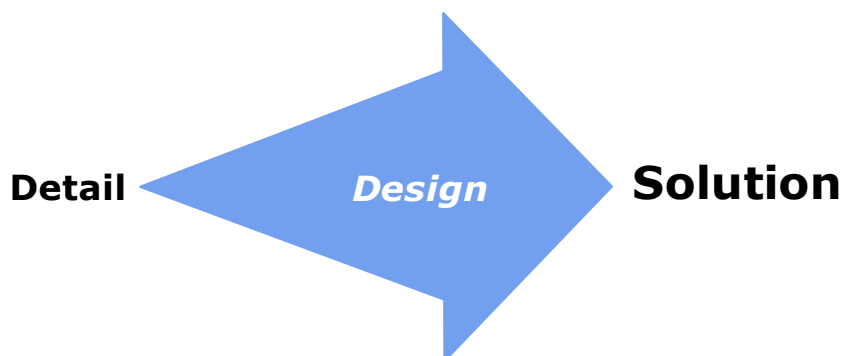
4) Analysis first, design second

As a business analyst we can often be caught between giving the business user what they want and knowing what the company is (or is not) able to deliver. Herein lies one of the greatest traps the analyst can fall into – designing the solution before they've understood the problem.

It is very difficult to gather information on requirements - without starting to design a solution. It is often hard to distinguish between a requirement and a solution as both look like similar functions. However we must stop and think about what we are doing in the context of the big picture. Analysis breaks something that already exists down into its components to so we can understand how it works (or doesn't work).



Design on the other hand builds up from the detail so that we can define a solution which does not yet exist. The processes are therefore opposites, but the deliverables can look much the same. It is what we do to produce the deliverable that matters. If we design before analysis is sufficiently complete, we may miss the real requirements, we may miss opportunities for a better solution, we may specify a flawed solution.



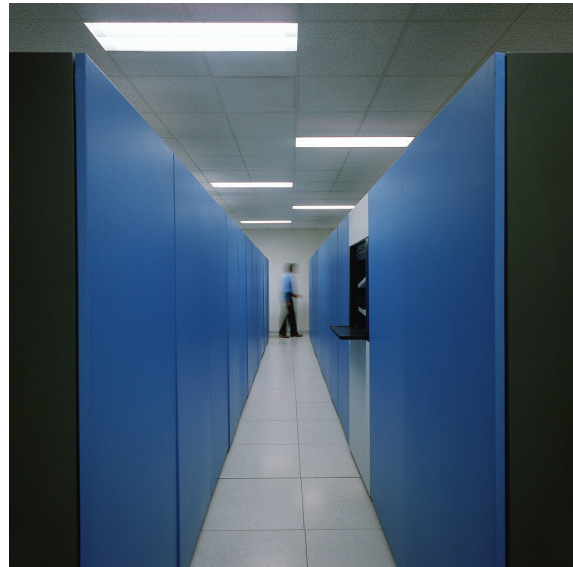
Case Study - The State of Florida planned to implement a new welfare administration system. The IT project team were instructed to re-use several million lines of code from an already developed system. Unfortunately, it was a centralised system whilst the requirements called for a distributed system. The difference in architecture prevented a successful implementation despite all efforts. Choosing a specific solution without considering the requirements, while merited on financial grounds, simply did not work.

5) What applies to small systems doesn't apply to large ones

This golden rule is based on the highly regarded 1976 publication by Frank DeRemer and Hans Kron³ which described how complexity grows exponentially the larger the system. Here's a modern day example.

With the advent of the home PC, home networks and the Internet, business users are well acquainted with the use of computers in various environments. They then wonder why the costs of larger corporate systems are so high. Business analysts must work to educate business users about levels of reliability, redundancy, recoverability - and the ensuing complexity.

Take data storage and RAID disc technology. RAID (redundant arrays of inexpensive disks) is now creeping into home networks but has been a staple of corporate systems for almost 20 years. A RAID disk looks to a system like one logical unit but is actually a cabinet containing many individual disk drives. By configuring these drives in different combinations (RAID 0,1,2,3...etc) we can have a choice of disk mirroring, data stripping or just using the full capacity of each disk.



Home users can now buy an affordable RAID system as a home server. However if it has 1TB total capacity with two disks inside and is configured as RAID 1 then you're only getting 500GB (half a terabyte) of usable storage as your data is being duplicated on each disk. Your real cost of storage is double your purchase price on a \$ per GB basis i.e. disk utilisation is 50% (usually less when you take into account system overheads).

In a corporate data centre it's typical to have different RAID configurations depending on the application. Some configurations are best for performance others for recoverability. What's important to note is that disk utilisation (or bang for buck) can vary from below 50% to almost 100%. When you have an online data farm approaching 100TB or more then costs - including management, backup, maintenance - are in a different league to home systems. Therefore as a business analyst you may have to explain to users the pros and cons of keeping all customer data online when, for example, discussing requirements for a new data warehouse (data mart) system.

Case study - A government department decided to re-develop all of its reports as data marts and a reporting database. They began gathering requirements by inviting personnel from every section and team to attend a presentation of what the project would provide. The people invited were not the ones who knew how the business processes in the department worked so they could not provide the information that the IT team required. They were however, able to state what they wanted from the new system.

³ Frank DeRemer and Hans H. Kron, 1976. *Programming-in-the-large versus programming-in-the-small.*

The presentation raised expectations throughout the department and increased scope. When the schedule was delayed and no news was heard from the project team, the users became disenchanted and critical with the project. Users did not understand how adding more requirements had such an impact upon resources, costs and delivery times. Each user had a computer at home and could buy software online and install it in minutes. They couldn't understand why the project team could not do the same. The system was eventually delivered successfully, but not to the user's complete satisfaction.

6) Don't allocate conflicting project roles to the same person

Project members will typically be given more than one role in a project team. But some roles have conflicting objectives. One of the most common mistakes is to allow a programmer to conduct all of the testing on their own programs. The programmer can do their own unit testing, but another person should test it again later. It's so easy to miss our own mistakes. How many times have you proof read something you've written yet missed the obvious mistake?

Another potential conflict occurs between product managers and project managers. The former represents the user and wants as much functionality in the application as possible. The project manager must ensure delivery of what is in scope, on target, on budget, of good quality, and must resist (without formal consideration) additional functionality.



Case study - A software provider of a configurable application used their project managers as client account managers. Clients inevitably tried to get the vendor to include new functionality. This usually came in the form of several small requests as the client slowly gained more experience of the software. The project/account manager felt under pressure to accept them. The project team operated under a constant stream of scope creep. The software provider's company culture accepted this as normal, but most of their profits were eaten up in freebies to the client. The client however, did not respect the provider for being so easy to dictate terms to. The provider lost even more respect when the application was delivered late because of the extra functionality. The provider was less profitable than they were expected to be and during some difficult financial times, were taken over quite easily and cheaply.

Summary

Methods, techniques, tools and trends will come and go, to be replaced by others. Some will be totally new ideas, others are repackaging of what has gone before - perhaps in a new form better suited to the times. Golden rules can be distinguished by the fact that they are timeless, consistent and can be applied in many industries and for a variety of systems and applications. Unfortunately they are often neglected or forgotten by those who know them, or missed by those who don't. But they are always there and always pay a good return on investment to those who use them.

Glossary

Six Sigma: a formal set of business management practices, originally developed by Motorola to remove defects in manufacturing and improve business processes. Now in widespread use around the world.

ITIL: a set of best practices for managing IT infrastructure and operations. Developed by the Office of Government Commerce in the UK and in widespread use.

COBIT: another set of best practices, this time for implementing IT governance and audit controls. COBIT was developed by ISACA an organisation for IT governance professionals.

Sarbanes-Oxley: a US Federal Law specifying minimum reporting standards for public companies in the wake of the Enron and WorldCom accounting scandals. Seen by many as the global, de facto, governance standard.

AIPM: Australian Institute of Project Management. Formed in 1976, has 8,000+ members.

PMI: Project Management Institute. Over 265,000 members in 170+ countries.

PRINCE2: a widely used project management methodology from the Office of Government Commerce (UK), the same body responsible for ITIL.

CBAP: Certified Business Analysis Professional qualification from the International Institute of Business Analysis.

QBAP: Qualified Business Analysis Practitioner certification from the Australian Business Analysis Association.

© 2009 IRM Training Pty Ltd. All rights reserved.

Send feedback and comments to: training@irm.com.au

You may use this article in your newsletter or internal document free of charge, provided that you do not alter it in any way and include all copyright notices.
