A post with the extreme long title: Trends transforming a CIO's approach to Legacy Modernization inspired me to write about the way I prefer to design systems for change. After listing all of today's trends that businesses are expected to work on sooner or later, the post concludes that the future is hard to predict;

> *we can't start an architecture by asking for the requirements*

and we should stop looking for a solution. The remainder of the blog is trying to make you want the next solution for this problem … we have heard that fairy tale before.

> *Do you think it is true that an unknown future is a new phenomenon?*

I don't think so. The future of today is as hard to predict as the future of two decades ago. System engineers in the past often advised businesses to fix the current practice under the illustrious title: business requirements. I suspect that this was often advised on purpose since:

> *the next change request would become the billable hours of the next year.*

Which salesperson would not like that perspective? The lack of proper system engineering skills may be another reason for the advice at a time when the demand for technically skilled staff was exceeding the supply.

> *There are simple ways to design systems for change and they have been around for ages.*

How else can we explain that some legacy systems are still running today? They (already) used (some) of these techniques. I will demonstrate how the method works for an information model that deals with the unknown future.

Describe the essence of the unknown future

We start by knowing for certain that now and in the future we would like to store information about 'something'. But, we don't know what information and we don't know what that 'something' is named. We all know by now that information can be stored in a table (think of Excel) and here is a simple example:

Property ID	Concept ID	Value
1	1	Name
2	2	City of birth

The above table provides a structure to store the idea that we have information about someone's name and birth date. There are two concepts in our table but it's easy to see how to extend this table with birthday, address and all the unknown information of the future … right?

> *The information in the table is easy to extend to all the unknown information of the future.*

The funny thing is that we can also store information about an actual case, like "Silvie was born in Amsterdam", in the same table.

Property ID	Concept ID	Value
1	1	Name
2	2	City of birth
1	3	Silvie
2	3	Amsterdam

Now that we know this trick we can even imagine how to store Silvie's relatives:

Property ID	Concept ID	Value
1	1	Name
2	2	City of birth
1	3	Silvie
2	3	Amsterdam
1	4	Paul
2	4	New York
1	5	Relative of
5	4	3

Paul is a relative of Silvie and born in New York. And so on and so on.

> *In a similar way we can deal with unknown rules and processes of the future.*

Design for the unknown future

Make a meta model that describes the essence and not the current reality. Define easy to use services on top of the meta model for easy querying and calculations. Most of the drawbacks of these design patterns are taken in by faster processors and distributed or parallel computing algorithms.

Do it better this time

Some of the legacy systems that are still running today use these extensible information models. That is why these systems survived. They are a pain for organisations because the rules that query and calculate the values are hard wired in code.

> *Don't make the same mistake again when running for the next 'solution'.*

As a CIO you should make it your top priority to use only systems that are able to deal with the unknown rules and processes of the future. The design is simple, the technology exists and performance barriers have been taken away.

Next tip: as a CIO you should invest in the new generation business owners such that they can adjust the rules and process on a whim. Shortage of programmers will remain an issue, but not for you.

Finally a CIO should understand the basics of software design to assess whether trending products are really designed for the unknown future or whether they are just the next legacy nightmare.

This is the most technology oriented post in a series to help shape the future of business driven software development. Continue reading on business rules in scrum projects and controlled natural languages.