Bristol

11 Rules for Specifying a Successful SCADA System

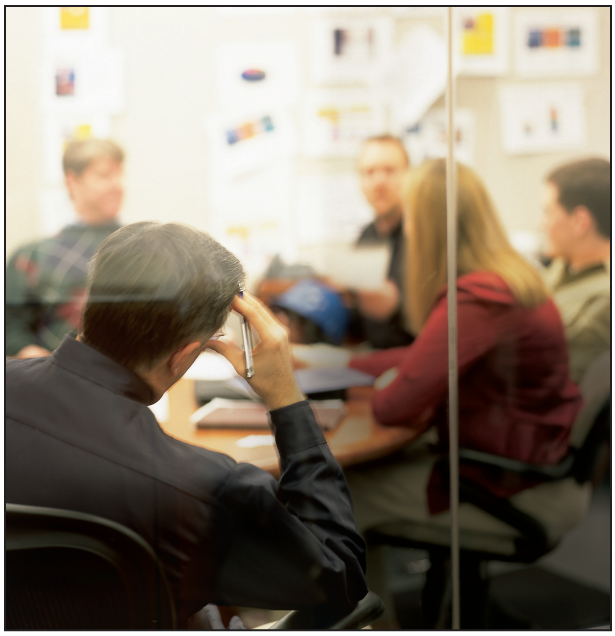
By Steve Hill

As a supplier of SCADA Systems and Software, we see a lot of specifications for SCADA systems. We also often help customers in determining their requirements as they develop specifications.

The specification is probably the most important part of your SCADA system – it determines the products and services you are going to receive, and it will be the reference against which you measure the final delivered product.

Based on our experience, we've gained a good insight into what makes a 'good' specification, and what is a 'bad' specification. A good specification will ensure that you get the functionality you require, at a reasonable price within a reasonable timescale. A poor specification may result in you getting something very different to what you expected, and paying far too much for it. A good specification will ensure that you have a small number of qualified bidders, all of whom can be reasonably expected to deliver a quality product. A poorly written specification may result in either no bids, or even worse, a wide range of (apparently) compliant bids from inexperienced or incompetent vendors. As SCADA moves into the world of IT the quality of the specification is even more important. Over 80% of IT projects worldwide fail to deliver, and the majority of these failures can be traced to poor specifications.

Over 80% of IT projects worldwide fail to deliver, and the majority of these failures can be traced to poor specifications.



Here's a series of basic rules that you should follow in developing a specification:

1. Assemble A Team of 'Users'

Those are the individuals who will interact with the system, maintain it or use the data provided by it. Try to keep the team at 5 or fewer members (perhaps by having a representative from each department, rather than the entire department).

2. Assign A 'Project Champion'

The team leader has the responsibility of delivering the final specification (and, if possible, the project).

3. Each User Independently Specifies Requirements

Users submit them, in writing, to the project champion. Having the requirements in writing avoids a long series of meetings that often fail to reach a conclusion. Realize that there will be compromise, unless you are prepared to accept a 'custom' system – and that is not to be recommended, for reasons of cost and future support.

4. Avoid Technical Conflicts

The project champion should then collate those requirements, and examine them very carefully for conflicts. For example, if there is a requirement that data is collected every 15 seconds, but there is also a requirement that existing 1200Bd modems are used then there is a potential (and likely conflict).

It is better to state the conditions that will apply, and to ask for the expected performance. For example:

'The system must display refreshed data for all 10,000 signals every 15 seconds. The vendor should include within their proposal for communications infrastructure to achieve this'

or

'The System will be communicating via the existing 9600Bd communications lines. These lines currently exhibit less than 1% communications failures. Please indicate the expected field to screen update rate if collecting 10,000 signals.'

In theory, any vendor would point out potential conflicts in their proposal, and state non-compliance.

Unfortunately, in many municipal bidding situations it's not possible for vendors to state non-compliances (it is assumed that all bids are fully compliant), and in any case vendors often avoid including non-compliances for fear of appearing to have a weaker solution than their competitors. If the project champion determines there is a conflict between requirements from different members of the team, he should work in small, targeted meetings to resolve those issues (rather than involving the entire team).

If you don't have the technical knowledge to determine if conflicts exist, contact the (potential) vendors before the specification is finalized, or speak to a consultant.

5. Tackle the System Availability Issue

One area that is always difficult is system availability. We are often asked to quote (and provide calculations) to 'prove' system availability. These requirements often arise in specifications that first saw the light of day over 20 years ago, where manufacturers were building all their own hardware and knew the MTBF numbers for every component.

Nowadays, almost all of the SCADA hardware will be based on 'off the shelf' components.

The specification is probably the most important part of your SCADA system – it determines the products and services you are going to receive, and it will be the reference against which you measure the final delivered product.

For many of these (especially PC hardware), it is simply not possible to obtain the availability figures from the manufacturer. Even if you could, a theoretical calculation is not of much use in determining how reliable the system is (because it's based on hardware reliability). A far better approach is to specify a typical system availability (e.g. 99.99%), and to specify what that means:

e.g. *'data collection and storage shall proceed with less than 1% bad data for 99.99% of the time, and 100% Operator workstation functionality shall be available at 90% of the installed workstations for 99.99% of the time. All reliability figures to be measured over a 12 month period'.*

Realistically, no vendor will be able to provide complete documentation to prove or disprove their compliance with these requirements – but you should be able to confirm these numbers when speaking to their references post bid. You can also measure these metrics post installation if you believe you have a reliability issue.

6. Don't Base the Specification on Your Existing System

We see many specifications that are written by users of legacy systems that are perhaps 10 or 15 years old. These specifications then refer to functionality that may simply be no longer available (because Operating Systems or hardware no longer support or require it). We sometimes see statements like:

'The Historical System shall support all features of the existing SupaSCADA '87 TLog Trending system'.

Instead of doing this, you need to analyze the functionality of the existing system, and specify the functionality you require, and which is easily identifiable by the vendor.

'The system shall be able to store up values for up to 100,000 Analog points, at a rate of 1 sample per minute for up to 10 years. Each sample shall be time stamped to a UTC timestamp with resolution and accuracy of 1 second or better.'

Sometimes you must refer to the system that is being replaced. In which case be sure to provide the data that the selected vendor will require to estimate the work required. For example:

'The system shall be able to import data from the .XYZ historical backup files of our existing SupaSCADA system. A copy of the format of these files, with example files can be made available to all bidders.'

You should be aware that even for some legacy systems details of protocols and file formats may be proprietary, and protected by copyright or licensing. If that is the case be sensitive to the fact that a vendor may not be able to use this information, or may even be required to pay licensing to the supplier of the legacy system. You should obviously ensure that your new system doesn't have the same restrictions.

7. Conduct Your Site Visits Early

Before starting on your specification, get demonstrations, and if possible, evaluation copies of the software you are considering. You must visit existing customers of the vendors you are considering (note – BEFORE writing the specification). See how they use the packages, and see the features you like (and don't like). See how these users interact with their systems (in many cases there are different ways of doing the same thing – for example, how is historical data handled long term?). We see many customers only doing site visits AFTER they've written the specification and received proposals.

A visit 'post bid' is very useful to ensure that prospective vendors have the required experience, and produce a quality installation on time, and are commercially viable – but are of little use in determining the functionality of the system you will end up with.

8. Avoid Getting Caught Up in "Look And Feel" Issues

We often see problems when customers specify interaction and behavior of the user interface (e.g. how a window is maximized or resized across multiple monitors, or how zooming and panning are performed). This type of thing is very different between Operating Systems, and even between different versions of the Operating System. It's often best to simply specify the Operating System(s) and Version(s) that you are prepared to accept – and indicate that no part of the operating system can be modified or replaced. Realistically, your acceptance of the 'look and feel' of a particular product is going to have to be based on a hands-on evaluation, and the site visits and conversations with existing users. Attempting to specify to a software vendor how their user interface should behave is a sure way to limit the number of bidders, or result in an expensive system. If there are a few small features that you

absolutely must have in terms of user interface (UI), then specify them – but make sure that all these features do exist in one product!

9. Pay Attention to How the Product is Configured

A surprisingly large number of products require the same information to be entered in multiple places (e.g. a tag name has to be manually typed as part of the RTU/PLC program; as part of the database configuration, a trend configuration and part of a report configuration). This will result in a system that is slow and difficult to configure, and also likely to contain many errors.

To avoid this problem, you should indicate that reference or use of any existing configured item shall not require all or any of that configuration to be reentered – it shall be possible to select from the existing configuration when re-using or referencing it.

As part of your evaluation of the software products be sure to configure it yourself. It is a good idea to make a 'script' of all the normal tasks you expect to do on the system (e.g. adding an I/O point, adding it to a mimic, a trend and a report) and have the vendor show you the entire process on their system. This is the only way you will discover if you can 'live' with the system – and discover the true limitations of the package. Pre-configured demo systems and evaluation packages often have acres of 'eye candy' – 3d graphics prepared by professional artists. Do not judge the system by these - judge it purely on the usability and day-to-day functionality, and in particular the time it takes to perform the standard day to day tasks (both as an administrator and an operator).

If there are features you know are in one product (and not others), but you'd like to see added, put these in a separate section in the specification and ask for an additional cost as an option.

10. Refine the Specification

Once you've seen a lot of different packages and installations, you'll begin to have a good idea of the system you require. You may even have a preferred vendor or package. You probably want to target your specification such that it specifies the functionality that is to be found in that product.

Depending on your organization you may be able to specify the product by name (preferred, if you have a genuine preference), or via a 'lock-in' specification. One thing to be very careful of (we see this in almost all specifications) is to make sure that you do not combine 'unique' features you've seen in different vendors products into a single specification! For example, Vendor A may have a very unique trending feature that you like, while Vendor B has a unique alarming feature. Putting both into the specification as 'required features' will probably result in each vendor having to include costs to try and match the other (and thus increasing your pricing). The result will probably be a late project, and perhaps architectural issues. If there are features you know are in one product (and not others), but you'd like to see added, put these in a separate section in the specification and ask for an additional cost as an option. In addition, ensure that you are aware of which required features are already in the product, and which are not. The simple requirement, *'state if this feature is available in a currently released version of the product'*, can help bring out development issues.

11. Finally, Determine How You Will Judge the Returned Proposals

Organize your specification carefully. Number each paragraph and requirement individually and require that the bidders respond using the same numbering system (it's a good idea to include a spreadsheet or table for them to complete). Internally assign a maximum 'value' (usually as points) to each requirement or required response. When you receive the responses you should then evaluate them based on 'marking' the response to each requirement and totaling the score. Obviously, price should also be given a value in this assessment. Each member of the team should 'mark' the proposal individually and without reference to the others – though any major discrepancies in opinion should be resolved in small, targeted discussions.

Specifying a Successful SCADA System

If you are in a 'lowest bid wins' situation (where the bids are assumed to be fully compliant) then you obviously can't evaluate the bids. In this case the next step of the process – project implementation – is the time when you can determine if you are going to get what you asked for. It's obviously important for ALL purchasers of complex systems to carefully monitor vendor progress through the implementation phase, but it is most important for these customers. In order to constructively monitor the vendor's progress, it's important to understand the process they use to design and implement your system. We will cover that in the next article: "Implementing Your SCADA System- On Budget and On Time".

11 Rules

1. Assemble a team of 'users'
2. Assign a 'project champion'
3. Have each user independently specify requirements
4. Avoid technical conflicts
5. Tackle the system availability issue
6. Don't base the specification on your existing system
7. Conduct your site visits early
8. Avoid getting caught up in 'look and feel' issues
9. Pay attention to how the product is configured
10. Refine the specifications
11. Determine how you will judge the returned proposals

Emerson Process Management
Remote Automation Solutions

Watertown, CT 06795 USA T 1 (860) 945-2200
Mississauga, ON 06795 Canada T 1 (905) 362-0880
Worcester WR3 8YB UK T 44 (1) 905-856950

Website: www.EmersonProcess.com/Remote

EMERSON™
Process Management