So how should one go about actually building the integrated sector? In fact does the development approach even matter? And the answer is that yes, the development approach matters a great deal.

The proper way in which the integrated sector is developed is in terms of an approach that can be called the spiral development approach (or sometimes called the circular development approach.)

In order to understand the spiral development approach, it is useful to understand a little about how development methodologies were created.

Before there were development methodologies, there was chaos when it came to development. Fig meth.1 shows that in the beginning of development there was chaos.

Then Ed Yourdon, Tom DeMarco and others came along and defined a methodological approach for systems development. This approach is generally called the “structured” approach. Fig meth.2 shows the advent of the structured approach that led the way out of chaos.

Perhaps the defining tool of structured analysis was the functional decomposition diagram. The functional decomposition diagram was one that showed how a large function can be broken down into its component parts.

One of the essential part of the structured approach was the data flow diagram. The data flow diagram, or the “dfd” was the definition of what data was to go from what process to another. Fig meth.3 shows the dfd.
The Spiral Development Methodology in DW2.0
By W H Inmon

Another tool found in the structured analysis tool box was a tool called a CRUD matrix. The CRUD matrix stands for Create, Read, Update, and Delete. The CRUD matrix is used to show on a system wide basis where data is created, used, and deleted. Fig meth.4 depicts a simple CRUD matrix.

<table>
<thead>
<tr>
<th>Data 1</th>
<th>data 2</th>
<th>data 3</th>
<th>data 4</th>
<th>data 5</th>
<th>data 6</th>
<th>data 7</th>
<th>data 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRU</td>
<td>CRU</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
</tr>
<tr>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
</tr>
<tr>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
</tr>
<tr>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
<td>CRUD</td>
</tr>
</tbody>
</table>

The process modeling exercise, which is another term for structured analysis and design, is then made up of tools such as functional decomposition, data flow diagrams, CRUD matrices, and so forth.

Fig meth.5 shows the different components of the process model.
From the different components of the process model was developed the SDLC – systems development life cycle. The SDLC was an early roadmap for development. The SDLC requires first one development activity to occur, then another. The SDLC is sometimes called the “water fall development” approach to systems development. Fig meth.6 depicts the SDLC.

When all put together, the structured analysis tools led to what can be termed the SDLC. The SDLC stands for the systems development life cycle. The SDLC shows the complete activities needed to build a system. Fig meth.7 shows an SDLC.
The SDLC is sometimes called a “waterfall” approach. It is called the waterfall approach because each level of development – once complete – feeds its output into the next level of development. The resulting flow of work product resembles a waterfall.

Fig meth.8 shows that one condition of the SDLC is that the requirements be gathered at the outset of development.

The condition is fine for operational systems where people have been performing the same work for a number of years. In the operational environment the work is repeated so often that it becomes rote. Under these circumstances it is entirely possible to determine requirements before the development process begins.

But in the case of DSS users – analysts – the requirements for processing normally are not known before processing begins. At least ALL the processing requirements are unknown. Fig meth.9 shows that there is a dilemma when it comes to gathering requirements when the requirements are not known.
And why is it that DSS analysts usually don’t know their requirements? It is because DSS end users operate in a mode of discovery. DSS analysts operate in a mode of “give me what I say I want, then I can tell you what I really want.” DSS analysts say – I can tell you my requirements as soon as I can see what the possibilities are. Fig meth.10 shows DSS analysts.

Fig meth.10
The community of DSS users - farmers, explorers, miners, and tourists

Fig meth.11 shows that DSS analysts operate in a mode not knowing what they want.

Fig meth.11
As a rule, DSS users do not know what they want. They operate in a mode of discovery.

DSS analysts must see the possibilities as a part of requirements formulation process. Fig meth.12 shows a DSS analyst searching for requirements.

Fig 10.12
Most end users operate in a mode of discovery - they don’t know what their requirements are until they see what the possibilities are.

Because of this uncertainty of requirements, there is a major disconnect between the SDLC and DSS analysis. The SDLC requires you to know all requirements before continuing the development process. The DSS analyst cannot know what the requirements are until at least some of the development is already done. Fig meth.13 shows the major disconnect between DSS analysis and the SDLC.
Because of this disconnect, a different approach to development is required. The approach that is required for the development of the integrated sector of DW2.0 is called the “spiral” approach. In some circles the spiral approach is called the circular approach or the iterative approach.

Fig. 14 depicts the spiral development approach.

In many regards the spiral approach is similar to the SDLC. But the spiral approach has one major differentiation with the SDLC approach and that difference is that the spiral approach operates on only a limited set of requirements. In stead of trying to gather all requirements before proceeding, the spiral approach accepts only a limited set of requirements. Then, the spiral approach develops the system just for those limited requirements. The development is done quickly and results are soon created. Then the spiral approach bites off another set of limited requirements and executes against those requirements. A new set of results are quickly created, and yet another iteration of spiral development ensues.

It is thus that the spiral development approach creates limited results very quickly.

Another perspective of the execution of the spiral approach is seen in Fig. 15.

Fig. 15 shows the totality of the requirements that come from the DSS analyst. Some of those requirements are known; many are not.
The first iteration of development ensues and some results are produced. This is depicted by Fig. meth.16.

Another iteration of development occurs and some new results are created. This is shown by Fig. meth.17.

Even more iterations follow and even more results are obtained. Fig. meth.18 shows the results that have been created after several iterations of the spiral development methodology has been executed.
One feature of the development life cycle is that it be executed to completion. And another feature is that the execution is done quickly. Fig meth.19 shows the rapid execution of any one iteration of spiral development.

The steps of the spiral development approach bear a strong resemblance to the steps of the SDLC. In fact the steps are essentially the same. Fig meth.20 shows the steps of the spiral development methodology.
The steps of the spiral development methodology are -

- requirements gathering
- analysis
- design
- programming
- testing
- integration.

One question that needs to be asked is – how do the iterations of the spiral development methodology become tightly integrated? After all, the steps of spiral development may be executed by different people over different time frames. The different outputs of development are coordinated because each development iteration operates from the same data model.

Fig meth.21 shows that the first iteration of development operates from a subset of a larger data model.

Fig meth.22 shows that the second iteration of development operates from the same data model, but from a different part of the data model.
The second iteration of the spiral approach uses another portion of the data model and produces some more of the data that is needed.

Fig meth.23 shows that the next iteration of spiral development operates from yet another part of the same data model.

The third iteration of the spiral development methodology uses a different portion of the data model and produces a portion of the data that is needed.

The final result is that the different development results fit together like a jigsaw puzzle because all the iterations of development have operated from the same data model.
Fig meth.24 shows the tightly integrated product that results from the many different iterations of spiral development. The final result is achieved iteration by iteration, one at a time.