# Mission-Based Incremental Development of C2 Systems for More Efficient Business Support

Ingmar Ögren
*Tofs Inc.*

*It is not possible to ever know enough about requirements before developing a Command and Control (C2) system. Also, a C2 system is never identical to its automated (computerized) parts. This article will introduce you to dominant battlefield awareness (DBA), how it is used to define a mission "Build DBA," and how abilities are connected to mission completion. Mission supports such as operator role, software, and hardware are used to model a complete C2 system, including four system levels: reality, computer information, presentation, and mental. The model is a useful basis for various simulations required during system development and evolution, as well as for acquisition of system components. Lastly, the possibility of extending the model to cover the domain of C2 systems is presented along with how that extension can become a basis for more rational production of C2 systems within the domain.*

The development and evolution of Command and Control (C2) systems have suffered from problems, some great enough to cause project failures. Two erroneous assumptions have contributed to this situation:

- That contracts can be based on the premise that a C2 system can be sufficiently known prior to development to produce complete and high quality requirements specification.
- That a C2 system is identical to the automated (computerized) support for such a system.

What is wrong with these assumptions?

We know from experience that new knowledge will be gained during C2 system development that will cause changes to the original specification. Areas affected include system usage, system environment, system features usefulness, new mission completion possibilities, etc.

The goal of every C2 system is to complete one or more missions through a combination of human operator tasks and automated support. Consequently it is important to understand how the system completes its missions and how to manage human and automated systems in the same context.

With this in mind, two better assumptions would be the following:

- Prior to developing a C2 system, identify its missions and expect the detailed requirements to surface during development.
- To complete their missions, a C2 system requires operators and cooperating software and hardware.

To fully understand these assumptions, we need an example.

## Dominant Battlefield Awareness

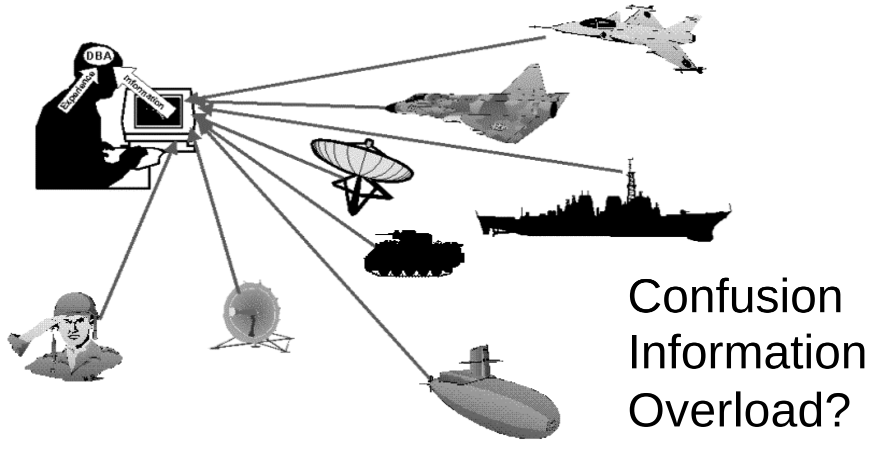
Dominant battlefield awareness (DBA) means that a commander in a C2 system builds an awareness of the battlefield situation by using many information sources such as agents, sensors, reports from other C2 centers, etc. Since DBA is an important part of the ongoing revolution in military affairs, the mission *Build DBA* is selected as an example to study system management.

Figure 1 shows that if all the information required to establish DBA in a real battle situation is linked directly to the commander two problems may surface:

- Confusion as a result of depending on data of different ages, from different origins, of differing quality, etc.
- Information overload through presentation of more data than is humanly possible to overview and understand.

The conclusion is that there is more to building DBA than to present all available information to the commander.

Figure 1: *Dominant Battlefield Awareness Requires Large Amounts of Information*



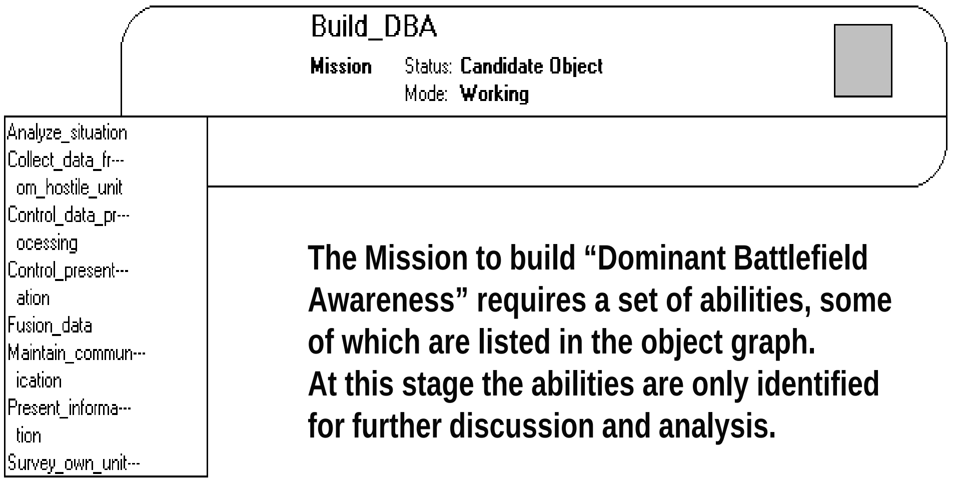
## Connectabilities to the Mission

Completion of the mission Build DBA requires a set of abilities. An approach defining the necessary abilities in connection with the mission object Build DBA is shown in Figure 2. The abilities listed are:

- Analyze the situation, including interpreting the data in the context of known behavioral patterns, for units in the battlefield.
- Collect data from hostile units, including sensor, agent, report, and other data about hostile units.
- Control data processing in building DBA to support the commander's objectives.
- Control presentation(s) of commander and others involved to support building necessary awareness.
- Fuse data to avoid double presentation and enhance information validity.
- Maintain communication ensuring that all other friendly units' information remains available.
- Present information database for the commander and others in a C2 unit.

- Survey individual unit so that information is presented in the battlefield context keeping the commander aware of the situation.

What is important is that the abilities required for mission Build DBA can be seen as actions offered by a mission object, and consequently drawn in an object graph as shown in Figure 1.



The Mission to build "Dominant Battlefield Awareness" requires a set of abilities, some of which are listed in the object graph.
At this stage the abilities are only identified for further discussion and analysis.

Figure 2: *Military Systems Operational or Being Developed Using the DII COE*

## Support Abilities with System Components

Now that we have defined a set of abilities, we need to create a system of components that actually has these abilities. In Figure 3, the mission object Build DBA is supported by a set of objects needed to build the abilities listed above. The support objects categories are operator (role), software, and hardware with examples:

- The "Commander" is an operator role object.
- The "Person-Machine Interface" and the "information base manager" are examples of software objects.
- "Data processing resources" is an example of a hardware object.

The diagram in Figure 3 is called a tree graph. It shows the need-lines between the objects in a system model. The version of the tree graph shown is drawn in the Tofs software tool, which also shows completion status for the different objects as little clocks.

## The Four Knowledge Levels

After building a system outline, you must consider what really constitutes DBA. The basic prerequisite is that the commander's mental awareness must comply with the actual battlefield situation.

Figure 4 (see page 28) shows the four knowledge levels necessary to understanding Build DBA, and which can be seen in a structured system model as shown above. The levels from the bottom are:

- The reality level representing battlefield reality or God's view.
- The computer information level representing all the data about the battlefield situation available in the C2 unit's computer system.
- The presentation level representing the information presented to the commander after data processing and selection.
- The mental level representing the commander's awareness after the presented information is combined with his personal experience and intuition.

It is obvious that the mental level must comply with the reality level to achieve DBA. The prerequisites for the required compliance can be studied in a system model built as a dependency structure as shown above.

## Incremental Development with Simulation-Based Acquisition

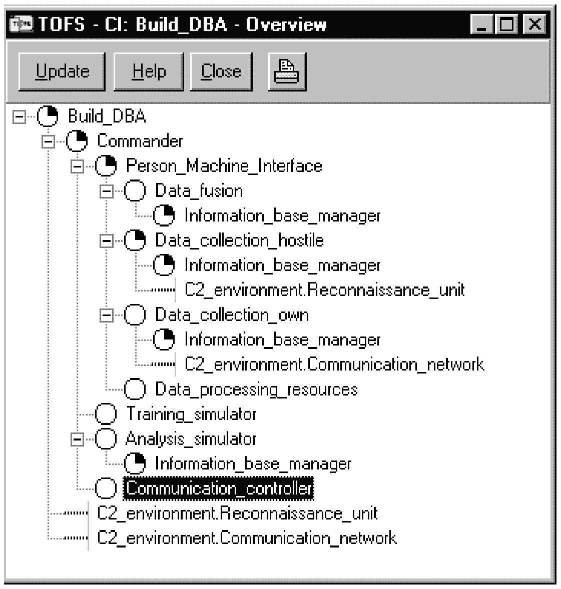
As stated above, it is not really possible to build a qualified C2 system from frozen requirements specification. Knowledge will inevitably grow during development, and some of this new knowledge will influence the requirements. Experience supports this insight since all non-trivial real C2 systems needed updates during development and repeated updates after the first delivery.

We conclude the need for an orderly way to manage changing requirements, to build and save new knowledge, and to change the system incrementally, especially as requirements insight grows and as the changing environmental situation introduces new requirements.

One way to do this is to use a model, as outlined above, as a system backbone. Use simulations to verify the model and investigate requirements, then let the model evolve incrementally. It will then be the system reference and a basis for both simulations and system component acquisitions.

Figure 5 (see page 28) shows a model used as a system reference, and how the model evolves through system increments. It illustrates how the project starts with an idea, how documentation can be connected to the model, and how the model is used as a basis for both simulations and system products (acquired components). Note that not only the system concerned needs to be simulated, but also its environment.

Figure 3: *A Model Outlined From the Mission Object "Build Dominant Battlefield Awareness" to Show the Components Needed*



```
Build_DBA
  Commander
    Person_Machine_Interface
      Data_fusion
        Information_base_manager
      Data_collection_hostile
        Information_base_manager
        C2_environment.Reconnaissance_unit
      Data_collection_own
        Information_base_manager
        C2_environment.Communication_network
      Data_processing_resources
    Training_simulator
    Analysis_simulator
      Information_base_manager
    Communication_controller
  C2_environment.Reconnaissance_unit
  C2_environment.Communication_network
```

**The model is built from the assumption that systems are best modeled using the relationship "depends on".**
**The components are of categories:**
- **Mission**
- **Operator (role)**
- **Software**
- **Hardware**

**The "clocks" indicate completeness status for each component (object).**

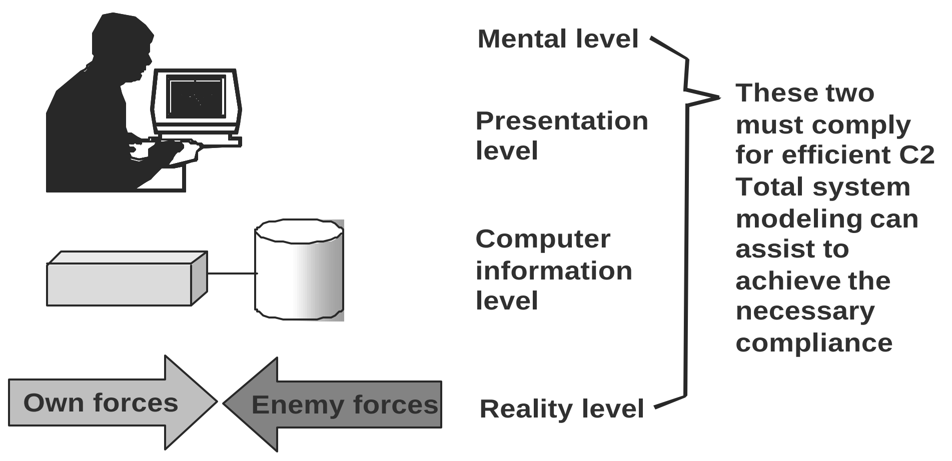

Mental level
Presentation level
Computer information level
Reality level
These two must comply for efficient C2 Total system modeling can assist to achieve the necessary compliance
Own forces
Enemy forces

Figure 4: *The Four Knowledge Levels and the Necessary Compliance*

## Formalize the Model

For a model to be a firm basis for multiple simulations and various system component acquisitions, it is essential that different system parts comply with each other and that the simulations comply with the system built. For example if the C2 system is of a non-trivial size, it must be possible to use a computer to check the model's consistency. The conclusion is that the model must be expressed in a formal syntax. Furthermore, this formal syntax must be readily understood by all those involved – developers, end-users, quality people, etc.

One way to achieve the required formality is to express the model in formal English using a limited language that includes:

- Reserved words to express control constructs.
- Variables of defined types.
- Comments that are used as explanations and to describe parts of the model that are not yet formalized.

The interaction between a commander and his Person-Machine Interface (PMI) to manage surveillance resources can be modeled as two concurrent processes. The commander's manual process interacts with the concurrent PMI software process through sending and receiving messages. An example of such a message is a presentation of a screen image that means the PMI has sent a message to the commander.

## Generalize the Model to the Domain Level

We have discussed how to use a model as a backbone for incremental development of C2 systems, and how it obtains compliance between system simulations and system implementations. Since the history of C2 systems includes some reinventing of the wheel, you may wonder: Can the modeling technique be used to avoid such reinvention? Models can be used in combination with so-called domain engineering to minimize reinvention. A possible principle is:

- A similar set of C2 systems are analyzed and used as a basis for a generic architectural model for the C2 application domain. The architectural model can then be formalized as described above.
- The analysis result is further used to identify reusable assets from the existing C2 systems. These assets are connected to the relevant objects in the architectural model to prepare for reuse. Assets may then be, for example, software modules, requirements, manuals, specifications, hardware products, etc.
- For each new or modified C2 system, you begin by combining the original requirements specification with the domain architecture to create a tailored version of the domain architecture to satisfy the specification (possibly modified after the analysis work).
- As far as possible, the new system is implemented using assets connected to the domain architecture. Some components will normally have to be developed anew. These new components may then be turned into reusable assets.

The result is an orderly development process that, provided a number of C2 systems with some similarities are to be produced, will decrease cost and development time and increase quality.

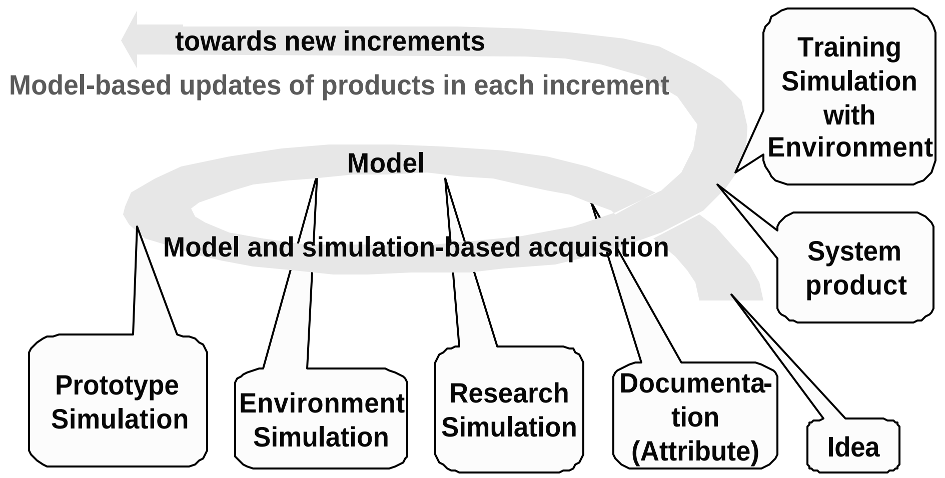
## Possible Objections

The principles described above may seem foreign and objectionable. However, if you take the Swedish Air Defense C2 system Stril 60, for example, it evolved from the early sixties into the nineties. This evolution was possible through cooperation within a group of technical and tactical experts with a good understanding of the system's missions, abilities, and structure. This suggests simply that the well-proven informal work, based on the informal understanding within a small expert group, can be supported by a formalized and commonly accepted system model.

There may still be objections to the model-based principle, but these are not too well founded:

- *We use specifications, not models for acquisitions.* Fine, but when you need more information than can be managed in a textual specification, why not supplement the written specification with a computer-based model?
- *You cannot model intuition, and our C2 systems depend on the experienced commander's intuition.* This may be true, but it can still be worthwhile to model all the routine details of the commander's interaction with systems for com-

Figure 5: *Using a Model for Iterative Development and Acquisition of C2 Systems*


towards new increments
Model-based updates of products in each increment
Model
Model and simulation-based acquisition
Training Simulation with Environment
System product
Prototype Simulation
Environment Simulation
Research Simulation
Documentation (Attribute)
Idea

puting and communicating in order to simplify such interaction. The result may be that the commander pays less attention to the support systems, consequently getting more time for his essential creative tasks.

- *You must separate tactical application development from technical acquisition.* Yes, this is traditionally what is done and may be one reason for experiencing problems with C2 system development. A working C2 system requires smooth cooperation between manual and automated parts. This smooth cooperation is best achieved through modeling the complete system as a single structure.
- *Computer-based modeling is just an expensive way to replace documentation.* The computer-based model will be a good basis for documentation and help make sure the documentation produced is really compliant with the system built or simulated. Using the model as a basis for documentation may consequently decrease documentation costs.

```
begin
     send Get_map(current_area)                              Commander's behavior
     receive Present_map(current_area)
     receive Present_surveillance_resources(available_resources)
     for resource in 1..total_resource  loop
          send Select_resource(resource_id)
          receive Present_adjustment(possible_adjustment)
          send Adjust(current_adjustment)
     end loop
end

begin
     while Commander_messages_received  loop
          receive Get_map(area)                              PMI software behavior
          Support_software.Get_area_map
          send Present_map(area)
          send Present_surveillance_resources(area)
          {The map information collected from the support software includes
          information about surveillance resources availabe in the area selected}
          receive Select_resource(Resource_id)
          send Present_adjustment(Possible_adjustment)
          {For each surveillance resource possible adjustment is received together
          with the map information}
          receive Adjust(Current_adjustment)
          Data_collection_own.Adjust_data_collection
          {The surveilance resource concerned is adjusted as required}
     end loop
end
```

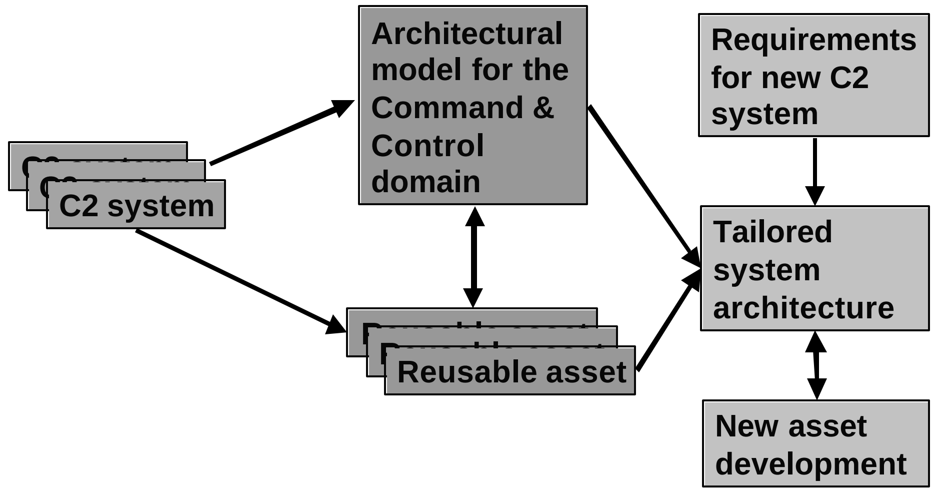
Figure 6: *Formalization of the Interaction Between a Commander and His Person-Machine Interface to Manage Surveillance Resources*

## Conclusions

Obviously C2 system evolution should start from a system's missions and abilities. It has been shown that model-based incremental technique for C2 development work has some advantages:

- Computer-based models can be used as a common base for simulations, acquisitions, recruitment, and training. While each system is too large for one person to overview, the model will help ensure that everyone involved knows his or her work is interconnected with the rest of the project through the common model.
- Models can help manage and structure large amounts of information that are traditionally stored as paper documents. This reduces cost and ensures that available documentation complies with the current system version or simulator. Using a model with good configuration management makes managing that information easier; relevant information for the current system version or simulator is extractable from the model.
- Models can be used to create a back bone in incremental development of C2 systems. Since the model lies behind each system version and simu lator built to support development and training, it is a backbone for the devel opment and reengineering work. This will help ensure that different system versions and simulators really comply with each other.

Figure 7: *Principles for Creating and Using a Domain Model*


C2 system
Architectural model for the Command & Control domain
Requirements for new C2 system
Tailored system architecture
Reusable asset
New asset development

In summary, model-based development and evolution means that well-proven principles are formalized and extended to cover larger systems with less dependence on expert groups.◆

## About the Author

Ingmar Ögren has a master's of science in electronics from the Royal University of Technology in Stockholm. He has worked with the Swedish Defense Material Administration and various consulting companies in systems engineering tasks associated with communications, aircraft, and command and control. He is currently partner and chairman of the board for Tofs Inc. and Romet, a systems engineering consulting company mainly utilizing method O4S. He also teaches systems and software engineering. Ögren is a member of Modeling and Simulation in Sweden and International Council Of Systems Engineering.

Tofs AB
Fridhem 2
S-76040 Veddoe, Sweden
Phone: (+46) 176-54580,
Fax: (+46) 176-54441
E-mail: iog@toolforsystems.com