# Assessing the impact of components on future military command systems

**Andy Ben-Dyke**
**Mark Cusack**
**Peter Hoare**
Distributed Technology Group
DERA Malvern
Worcestershire
WR14 3PS UK
+44 1684 894000
{adb, cusack, peteh}@signal.dera.gov.uk

**Mark Lycett**
**Ray Paul**
Department of Information Systems
and Computing
Brunel University
Uxbridge, London, UK
+44 1895 203397
{ray.paul, mark.lycett}@brunel.ac.uk

**ABSTRACT**
The future battlespace will consist of sensors, decision-makers and weapons systems interoperating within a framework of command systems. These command systems should be adaptable to meet the requirements of a specific operational environment and to take advantage of new technology. The barriers to achieving flexibility and interoperability within the military domain are analogous to those that complicate business-system integration within the commercial sector. Component software technology could play an important role in achieving the required level of flexibility and interoperability. We therefore describe a generalized framework for assessing the impact of component technology upon a system of systems.

**Keywords**
Software components, system of systems, assessment framework, quality characteristics, service-based architectures.

## 1 INTRODUCTION

Information Technology (IT) is having a dramatic effect on the way the military works, by providing support at the level of an individual combatant, right up to strategic decision-makers in central government. As well as the contribution that IT can make at any one level, the real benefits for computerisation come from the integration of IT between these levels, to support information flow up and down the military hierarchy.

The IT systems used by the military to co-ordinate and control operations are termed Command and Information Systems (CIS). This term is broad in scope, encompassing all aspects of computerisation, from off-the-shelf office automation, to specialist bespoke applications.

Current CIS consist of multiple stove-piped systems, where the connection between systems in the vertical chain of command is specific to a series of IT applications. These CIS are integrated together in isolation from any external systems, with connections only existing between applications within a single armed service. A system of systems then evolves from the creation of ad-hoc links between CISs. These links arise from the users' need to exchange information between different stove-pipes. Because there is no commonality between the applications, each link is unique and is therefore expensive to build and maintain.

The UK Ministry of Defence (MOD) is promoting a systematic approach to CIS interoperability, in which future systems are composed from a common set of service components. The principal focus for this initiative is the Joint Battlespace Digitisation (JBD) programme [1]. The aim of this programme is to achieve *information superiority* over an adversary through the integration of information from across the battlespace. Key to this is application-level interoperability between systems.

JBD defines an *aspirational future* in which more dynamic styles of application will form the basis of CIS. The phrase used to describe the organisation of this aspirational future is a *golf-bag of services*. This implies that services can be selected and organised as appropriate for a specific operational environment.

In reaching this aspirational future MOD faces a number of problems, including:

- Extremely long time-scales of IT procurements compared to the rate of technology change, resulting in equipment obsolescence before coming into service;
- Requirements changing between specification and the system in-service date;
- The lack of a coherent upgrade path as technology

evolves;

- The need for compatibility with commercial systems, to lower costs and to take advantage of new technological developments.

The challenges faced by the JBD programme are analogous to those faced by businesses when they merge and/or rationalise corporate IT systems. Such systems are often business specific, with limited external connections to ensure commercial security. However, the situation is changing with, for example, the drive towards B2B, B2C and Enterprise Application Integration (EAI). Hence there is a move to integrate these stove-piped systems using an open style of Internet applications and third-party implementation libraries.

**Defining the future requirements for CIS**

By the nature of the military procurement process, it is unrealistic to mandate a single approach for future CIS. Instead, requirements are framed in abstract terms that define the *qualities* of the CIS. These may include subjective terms such as *flexibility*, *security*, *reliability* and *cost-effectiveness*. It is generally agreed that the key qualities of future CIS will be flexibility and cost-effectiveness. Such qualities are often attributed to component-based software. However, this is a new technology, and it is important that the military understands the ramifications before adopting a component-centric approach.

In the following section we describe a framework for assessing the impact of component technology within systems of systems. JBD is used to develop and illustrate our assessment framework. It should be noted that this framework may be applicable to commercial IT systems of systems.

## 2 THE ASSESSMENT FRAMEWORK

**Motivation**

The main aim of the framework is to provide a systematic approach to determining the impact of components on a proposed system of systems. The term *system of systems* is used in this context to describe a set of individual systems, which although potentially operating independently, can also work as a whole.

The impact of adopting component-based software is measured in terms of the strengths and weaknesses of current component technology relative to traditional monolithic designs. Any assessment should also clearly identify areas for future component research, and include estimates of the potential benefits to a system of systems.

Ideally, the assessment should work at an abstract level and not require a detailed specification of the system of systems. Typical abstractions could include cost, efficiency, and flexibility. Furthermore, the framework should also allow multiple assessments using different prioritisation of the underlying properties of the system of systems. For example, one assessment may use cost as the overriding concern, whereas another may favour robustness or reliability.

We are not aware of any alternative approaches to that outlined in this section.

## 3 CHARACTERISING SYSTEMS OF SYSTEMS

This section describes the process of generating a set of metrics with which to assess the impact of components on a generic system of systems. Rather than starting in a vacuum we decided that we needed to look at typical command systems and identify their salient characteristics. Once these characteristics had been identified they could be fed into a requirements framework which would allow us to assess the effect of components on the system.

The approach of identifying the characteristics can be broken down into three phases:

- examine a typical command system as a case-study to extract important characteristics and properties;
- generalise the outputs of the first phase to apply to generic system of systems;
- re-assess the exemplar system against the generalised metrics to determine suitability and completeness of the metrics.

**Phase 1: The case-study**

By analysing the types of requirements described in JBD [1] it was apparent that the assessment metrics could be divided into two categories:

- **Service Characteristics.** System of systems can be described in terms of a service-based architecture. Within this framework, each service can be characterised in terms of the functionality it provides. Example services include communications and information and service management;
- **Quality Characteristics.** The intrinsic properties that are not embodied in any one service but are an accumulation of "value" across the whole system of systems. Typical characteristics include flexibility and cost.

To structure an assessment, each service and quality characteristic was partitioned into a number of sub-characteristics. For example, communication services were evaluated with regards to topology, bandwidth, etc., and cost separated into initial and through-life costs.

The service characteristics represent a technological view of a system of systems, and as such, assessment in these terms is straightforward. This analysis then provides the terms of reference within which to measure the quality characteristics. Note that the quality characteristics are sufficiently abstract to make direct comparisons against

technology both difficult and highly subjective.

**Phase 2: Generalising the metrics**
The second phase involved refining the quality characteristics with respect to the ISO 9126 standard [2], and generalising the service characteristics. The final service characteristics are defined as follows:

- **Application and Information services.** The producers, consumers, and manipulators of information and data. Examples include planning tools, reporting tools, command applications, GIS web browsers and weather servers;
- **Process support.** The manipulation of the Application and Information Services to aid the command/business process. Examples include workflow support, portals, decision desktops, consistent user interfaces, collaborative working and visual programming;
- **Data management.** The provision of the specified data to the requesting location efficiently and in the correct format. Issues include data replication, communication resource management, data distribution and routing;
- **Communication services.** The point-to-point transfer of data. Examples include email, wireless communications, telephone and post;
- **Service management.** The mechanism of implementing and enforcing the policies of the system by controlling the underlying services. Issues include user profiles and rights, system tuning, service availability, service installation and maintenance.

The quality characteristics are defined as follows:

- **Flexibility and supportability.** Flexibility is the *user's* perception of the system's ability to accept change. Supportability is the *system manager's and supplier's* perception of the system's ability to accept change. Issues include the introduction of new functionality, reconfiguration of functionality to match a new process, realising new functionality by combining existing functions, technology insertion and the removal of functionality;
- **Security.** The provision of unimpeded legitimate access to information while impeding illegitimate access. Examples include user identification and authentication, access control lists, encryption and decryption, security auditing and security labelling;
- **Efficiency.** The relationship between the level of performance and the amount of resources used, under stated conditions. Examples include time taken to start applications, time taken to perform a predefined processing task, HCI responsiveness and time taken to switch between applications;
- **Reliability.** The capability of the system to maintain its level of performance under stated conditions for a stated period of time. Issues include resilience during planned reconfiguration, fault tolerance through data replication and hardware/software redundancy;
- **Cost.** Cost is a measure of the resource usage through the system's lifetime, including people and equipment. Issues include the cost of procuring the system and cost of maintaining the system.

**Phase 3: Re-assess the exemplar CIS**
Having obtained a general set of metrics, we re-assessed the JBD exemplar. This provided an opportunity to refine the assessment framework.

## 4 THE ASSESSMENT PROCESS

Having established the metrics on which to base the analysis, the assessment process itself is broken down into five steps:

1. Assign priorities to the quality characteristics;
2. Provide a working definition for components;
3. Assess the system of systems in terms of the service characteristics;
4. Measure the quality characteristics of the system of systems (using the outputs of step 3);
5. Aggregate the results.

**Step 1: Prioritising the quality characteristics**
The prioritisation has a major impact on all of the remaining stages, but is necessary only when dealing with a system of systems for which there is no detailed technical specification. For example, when assessing the service characteristics, tensions between the quality characteristics will be revealed in the system-of-systems' design. In such cases, the prioritisation is used to resolve these conflicts and ensure consistency throughout the evaluation. As another example, the prioritisation will influence the selection of a component's attributes when generating the component definition — run-time interrogation of a component's interface may well be unimportant if cost or efficiency are the overriding goals. The full impact of the prioritisation on the final assessment will be inversely related to the detail of the specification of the system of systems.

**Step 2: Defining components**
At present, no standard definitions of components or component-based software engineering exist. To illustrate this, consider the following four definitions used during a workshop at the 11th International Conference on Software Engineering [3]:

1. A component is a non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture. A

component conforms to and provides the physical realisation of a set of interfaces;

2. A run-time software component is a dynamically bindable package of one or more programs managed as a unit and accessed through documented interfaces that can be discovered at run-time;
3. A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party;
4. A Business Component represents the software implementation of an 'autonomous' business concept or business process. It consists of all the software artefacts necessary to express, implement and deploy the concept as a reusable element of a larger business system.

Obviously, the definition used will have a major impact on the outcome of the assessment. Furthermore, the definition should include relevant extensions based upon future software component research. For example, the definition used in the JBD case-study (see section 5) admits the possibility of component wrapping and dynamic re-configuration. This allows the evaluation to include an assessment of the future impact of components on the system of systems.

**Step 3: Service characteristics**
The service characteristics provide a technical description of the system of systems. Using the sub-characteristics to provide structure, the potential roles of components in realising the specified functionality should be enumerated. For example, during the assessment of data management, it was determined that components could be used within the *formats* sub-characteristic as follows:

- For manipulating different data formats;
- Extending the range of formats by introducing new components;
- Improving the number of format conversions by replacing components;
- Transparently mapping complex data, e.g. compressed, secure JPEG image, by component wrapping.

Note that this assessment is driven by the goal of achieving the maximum overall flexibility, as determined by the prioritisation of the quality characteristics. Schemes where components offer no tangible advantage over traditional approaches should be ignored.

This stage of the analysis requires a sound understanding of component-based software engineering as well as a solid grounding in traditional software engineering. However, care should be taken to only make use of attributes from the component definition developed in step 2. Moreover, it is possible that potential research areas may be identified during the assessment. In such cases, the component definition should be extended to include this possibility, assuming that the benefits are in-line with the priorities established in step 1. If the component definition is modified, then the assessment should be re-started to ensure the full implications of the extension are captured.

**Step 4: Quality characteristics**
Measuring quality characteristics involves pair-wise combinations of all of the service- and quality sub-characteristics. For each pair, the assessment should be based upon: the technical assessment of the service sub-characteristics; the definition of the quality characteristics; the component definition; and any supporting external evidence. Due to the subjective nature of this assessment, care should be taken to provide sufficient details as to how components impact on the quality sub-characteristic. As an example, the assessment of cost in terms of data management was as follows:

- Increased start-up costs due to the design of the data-manipulation components;
- Reduced through-life costs as it is easier to add new formats and conversions.

**Step 5: Summarising the results**
The final step of the process involves the aggregation of the quality sub-characteristics, to obtain a final assessment of the impact of components on each quality characteristic. If the component definition from step 2 includes potential extensions due to future component research, the aggregation will be in terms of both the current and future impact of components.

For each pair, the impact of components is determined to be positive, negative, or neutral. Then, an aggregation is made across the services. For example, components may lead to a positive benefit in terms of the functionality of the application and information services. For each quality characteristics, the services are then ordered in terms of their relative impact. This is then used to obtain the final measure for the quality characteristic.

## 5 APPLYING THE ASSESSMENT FRAMEWORK

Applying the process described in the previous section back to JBD, we obtained the following outputs.

**Step 1: Assign priorities to the quality characteristics**
For the purpose of this assessment, the following prioritisation of quality attributes was used (based on a number of internal workshops):

1. Flexibility and supportability;
2. Security;
3. Reliability;

4. Efficiency;
5. Cost.

**Step 2: Provide a working definition of components**
Within the scope of the JBD assessment, components are assumed to possess a number of properties. A component should be non-trivial, in that it should have a minimum granularity and a clear function. A component should be nearly independent, only requiring a component framework and an explicit set of environmental dependencies. Each component should implement a well-defined set of interfaces and allow run-time interrogation of these interfaces. Finally, a component should be dynamically bindable.

These properties mirror the capabilities of current component technology. In addition to these, we wish to consider the following properties of a future component framework.

- **Dynamic re-configuration.** The safe updating of a components parameters;
- **Dynamic replaceability.** The safe insertion and removal of components at run-time;
- **Component wrapping.** The interception of component interactions to modify the functional and non-functional characteristics;
- **Semantic reflection.** The ability to reason about the non-functional characteristics of a component.

**Step 3: Assessment of service characteristics**
In this step, the results of the assessment of the impact of components on the system-of-system services are presented. The sub-characteristics of the services are described in terms of the definition of components given in the previous section. For brevity, we only include the service characteristic assessment for *process support*:

- **Understandability**
  - Strong likelihood of a direct mapping between the decomposition of a military process into task fragments and a service into software components;
  - Initially, services will be large-grained compositions of components, however, with time, services and components will become synonymous.
- **Accurateness**
  - Testing could be improved through component wrapping;
  - Potential for future improvements via semantic reflection.
- **Operability**
  - Interface reflection can help improve fragment selection by identifying compatible types in the functional definitions of two services;
  - Semantic reflection will improve fragment selection;
  - Components lend themselves to visual presentation and manipulation, and there is widespread tool support.
- **Adaptability**
  - Components are by definition composable and will therefore support process fragment composition. (In our scheme, a process fragment represents part of a business/command process represented by a software component.);
  - Dynamic binding and interface reflection provide a good platform for adding new process fragments.
- **Interoperability**
  - Components have no impact on the sharing and distribution of process fragments. However, the corresponding assessment for *application and information services* indicates that components are more portable.

**Step 4: Quality Characteristics**
For brevity, only the pair-wise combination of *reliability* with *application and information services* is included here:

- **Functionality**
  - Functionality is realised by the interaction of components. Component-Based Software Engineering (CBSE) is a relatively new technology, therefore stability may suffer. However, as experience with the technology increases so too will stability;
  - The use of interface reflection and dynamic binding may increase the automation of testing;
  - The use of wrapping may improve stability by supporting the introduction of additional safeguards;
  - Semantic reflection may allow comparisons to be made between the desired and actual behavioural model of a component.
- **Configurability**
  - Since all components make use of the same configuration mechanism, stability may be greater through increased programming experience and extensive testing of this single mechanism;

- **Operability**
    - Since all components make use of the same mechanism for achieving operability, stability may be greater through increased programming experience and extensive testing of this single mechanism.
- **Portability**
    - Explicit component dependencies ensure that the likelihood of an error occurring during the porting process is smaller, thereby increasing the stability.

**Step 5: Aggregation of results**

For this particular analysis, the summary is presented in table I. Each entry contains a rating of the impact of current and future component technologies. Future trends in component technology are indicated in brackets

From table I, it is evident that the aspects of JBD that most benefit from the application of components are flexibility, supportability and cost. Conversely, it can be seen that components currently have a negative effect on security and service management.

We believe it is likely that future research will address those aspects where current component technology fails to satisfy the JBD aspirations. In the main, this research will be driven by the commercial sector. Possible exceptions are security and service management, where the military's needs and priorities diverge from those of business.

## 6 SUMMARY

In this paper we have presented a framework for assessing the impact of software component technology of a general-purpose IT system of systems. To illustrate the framework, a case-study based on the UK MOD's JBD was presented.

This case-study indicated that components would have a positive impact on flexibility, supportability and cost of JBD. Conversely, the results show that current component technology would have a negative effect on security and service management within JBD.

One of the main problems that had to be addressed by the assessment was the fact that our example system, JBD, is so far only a concept, and its functionality and actual implementation are not yet defined. When assessing the role of components in such a situation it is difficult to make progress without making some assumptions about detail. However, by considering the impact of the technology on the *solution space* rather than looking at specific example systems, we have been able to make a systematic assessment of how components will affect a system of systems.

One issue raised by the assessment was that in making assumptions about the solution it is possible to bias the review unintentionally. In this assessment there has been a deliberate attempt to assess the ability of components to deliver flexibility. Starting with different input biases will obviously affect the outcome of the assessment (e.g. biasing towards cost may rule out certain technical solutions).

The initial finding of this study is that a component-based approach may offer significant advantages when realising a JBD-like system of systems.

**Future work**

An important area of future research is to assess the impact of the component definition on the assessment outcome. For example, the definition of a component can be considered at many different levels, from a low-level software component e.g. a GUI widget, all the way up to a *process* component embodying a business process or procedure. Starting with a number of definitions would produce a number of different results depending on the input terms.

Another area of future work is to broaden the assessment to other systems of systems, with the aim of validating the framework and better understanding the ramifications of component technology. If enough commonality was found between systems, it might be possible to develop a common approach to assessing any military system, or even commercial IT systems.

## REFERENCES

[1] Joint Battlespace Digitisation System Concept: An Introduction Issue 1 28th Feb 2000

[2] Information Technology - Software product evaluation - Quality characteristics and guidelines for their use, International Standard ISO/IEC 9126: 1991 (E)

[3] An Examination of the current state of CBSE: A report on the ICSE Workshop on Component based software Engineering (CBSE), Alan Brown and Kurt Wallnau. 20th International Conference on Software Engineering (ICSE) Kyoto, Japan April 1998

| | Flexibility and Supportable | Security | Reliability | Efficiency | Cost |
|---|---|---|---|---|---|
| Application and Info. Services | +ve | -ve (to +ve) | -ve (to +ve) | -ve (to neutral) | +ve |
| Process Support | +ve | -ve (to +ve) | -ve (to neutral) | -ve (to neutral) | +ve |
| Data Management | +ve (marginal) | neutral | +ve (marginal) | -ve (to neutral) | neutral |
| Communications | neutral | neutral | neutral | neutral | neutral |
| Service Management | -ve (to +ve) | -ve (to +ve) | -ve (to +ve) | -ve (to neutral) | -ve (to +ve) |

**Table I : Results of the JBD assessment**