

# Tutorial: Work Domain Analysis

---

Gavan Lintern

Cognitive Systems Design

[glintern@cognitivesystemsdesign.net](mailto:glintern@cognitivesystemsdesign.net)

Copyright © 2011 by Gavan Lintern

## Concept

Work domain analysis identifies the functional structure of a socio-technical system. That functional structure will encompass properties ranging from object descriptions, through specific and general functions, to values and system purpose. It identifies functional properties that result from design intent but in addition, functional properties that may not have been intended but instead were discovered by operators, and both desirable and undesirable functional properties generated because of interaction with context or environment.

Work domain analysis identifies structure independently of activity. It can be likened to a map that lays out the structure of a geographic environment. Activity is important, but neither a work domain analysis nor a map addresses that. However, both provide important leverage into planning of activity by laying out the resources available for action and the constraints on action.<sup>1</sup>

This approach might be clarified by consideration of the meaning of the terms *function* and *process*. These are troubling terms in engineering and science because their range of usage is broad and they have overlapping meanings. Within cognitive work analysis, Vicente (1999) has given them constrained meanings that map onto the needs of this analytic framework. A *function* is a structural property of the work domain. A *process* is a mechanism by which the behavior of the system is produced. This distinction is unusual and no other strategy of cognitive analysis makes it explicit. An underlying assumption of cognitive work analysis is that the separation of structure from activity helps bring an important source of order to the analysis of complex, socio-technical systems.

The product of work domain analysis is an abstraction-decomposition space; a two-dimensional matrix that distributes functions across levels of abstraction (object descriptions, physical functions, domain functions, values and priorities, and domain purpose) and across degrees of decomposition. By convention, abstraction is represented on the vertical dimension and decomposition on the horizontal dimension.

A major contribution of work domain analysis is that it identifies means-ends relationships between functions at different levels of abstraction. A means-ends relation reveals the functions at one level that must be used for satisfaction of a function at a higher level. In most cases, a constellation of resources or functions at the lower level will be required to satisfy any function, value or purpose at a higher level. Even within Systems Engineering, where this sort of distinction would seem to offer an advantage, Functional analysis as discussed in many texts and reports (e.g., as in Blanchard and Fabrycky, 2006) is a functional flow analysis (essentially a process analysis).

In this discussion of means-ends relations, the reference is specifically to resources that will be used to

---

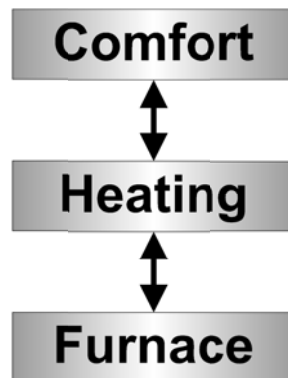
<sup>1</sup> Work domain analysis is part of the larger analytic framework of cognitive work analysis in which later stages that reference the analytic product of work domain analysis deal with activity

satisfy a functional requirement. It is often said that means-ends relations describes *how* a function is achieved but the word *how* can imply a reference to both resources and activity. In principle, a means-ends relation could specify either and it might even be useful under some circumstances to have means-ends relations specify both. However, the standard approach to work domain analysis specifically excludes any form of reference to activity and so a means-ends relation refers only to the resources that are available to achieve ends (which is actually consistent with the accepted definition of a means test). You should note this carefully because it is a source of considerable confusion within discussions about work domain analysis.

The remainder of this brief tutorial will focus on how an abstraction-decomposition space should look and hints about how to construct one. Note however, that the construction of an abstraction-decomposition space requires considerable knowledge about the system under consideration. The assembly of that knowledge constitutes a major knowledge acquisition effort, typically an extraction of relevant details from document reviews and discussions with subject matter experts. There is little in this tutorial that tells you how to do that but if you need further advice on that aspect of work domain analysis, I can suggest sources ([glintern@CognitiveSystemsDesign.net](mailto:glintern@CognitiveSystemsDesign.net)).

## The Abstraction-Decomposition Space

The concept of abstraction is depicted in Figure 1. In this example, comfort is the most abstract function and is enabled by heating (a physical function), which in turn is enabled by the furnace (a physical object). An important aspect not depicted in Figure 1 is that other sorts of resources not depicted here could enable both comfort and heating. The same abstraction, extended in Figure 2, depicts decompositions that might be useful for the analysis of a heating system.



**Figure 1: A simple depiction of an abstraction relationship with means-ends links**

The knowledge representation for work domain analysis, the abstraction-decomposition space, provides the foundation for the design of a radically new system form. This space represents functional properties of the work domain (objects, resources, constraints, values, purposes) in a two-dimensional matrix (Figure 3) each node represents a function. The vertical dimension represents the dimension of abstraction and the horizontal dimension shows varying levels of decomposition (system, unit, component, part).

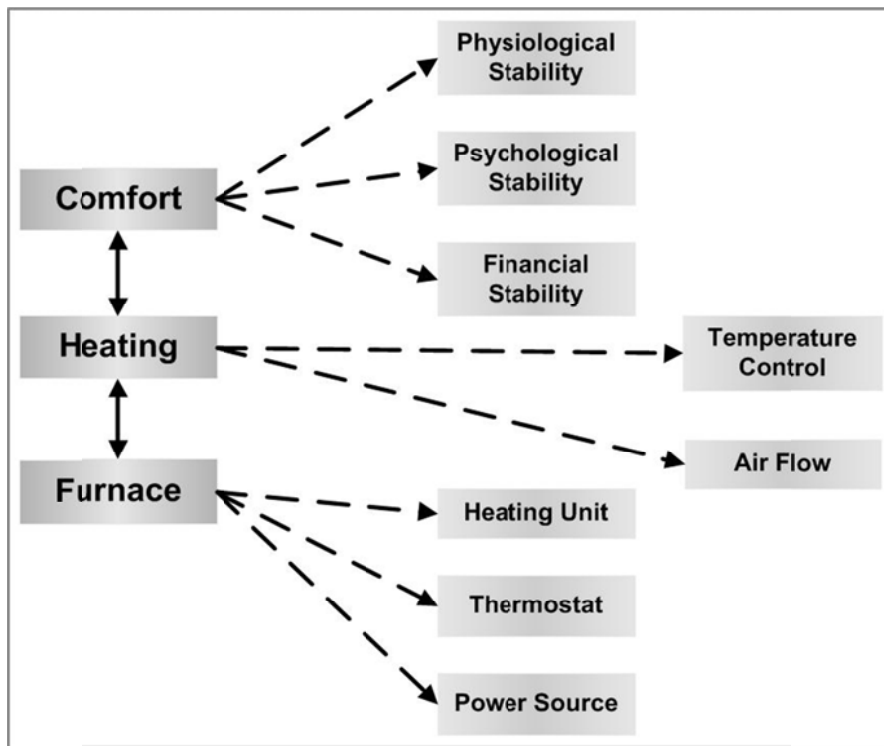


Figure 2: A depiction of decomposition relationships, building on the abstraction of Figure 1

| Decomposition Abstraction                 | System | Unit | Component | Part |
|---|--------|------|-----------|------|
| Domain Purpose                            |        |      |           |      |
| Domain Values & Priorities                |        |      |           |      |
| Work Functions                            |        |      |           |      |
| Technical Functions                       |        |      |           |      |
| Physical Resources Material Configuration |        |      |           |      |

Figure 3: The standard two-dimensional format of an abstraction-decomposition space

The abstraction dimension of an abstraction-decomposition space is typically organized proceeding from top to bottom) through the hierarchy of:

- **Domain Purpose;** the particular purpose or mission that is the focus of analysis
- **Domain Values & Priorities;** functions that encapsulate human and social values (e.g., safety-productivity trade-offs, concerns about collateral damage, conservation concerns with regard to own personnel and resources) and thereby constrain the space of acceptable action
- **Work Functions;** the general functions that will satisfy the domain purpose at a device-independent and purpose-related level of description
- **Technical Functions;** the effects or processes supported by or generated by technical systems or objects
- **Physical Resources & Material Configuration;** the physical properties of objects and devices such as location, layout, appearance and shape with special reference to properties that constitute resources or constraints for realization of technical functions at the next level up

Further discussion of the meaning of these labels for the levels of abstraction are presented in appendix A to this tutorial.

Abstraction levels are connected by means-ends links, shown in Figure 3 as solid, two-headed arrows. This two-headed arrow is used to indicate the reciprocity between related functions at different levels; a function is enabled or supported by functions to which it is connected at lower levels (its means) and conversely, a function is implemented to support or enable functions to which it is connected at higher levels (its ends). Decompositions within levels are represented by single-headed, dashed arrows with the arrow pointing in the direction of the decomposition.

You might need to deviate from the standard representational form for the abstraction-decomposition space because the multiple-column format cannot easily be made legible in a normal-size document page. An alternate form, shown in Figure 4, relies on dashed arrows to indicate decompositions within abstraction levels and conversely, enclosures to indicate aggregations. A decomposition of system into units is shown at the abstraction level of work functions and a decomposition of a unit into components and an aggregation of components into unit are shown at the abstraction level of technical functions. Decompositions should continue to the level of operational relevance.

Figure 5 offers a tutorial example of an abstraction-decomposition space that shows some functional elements for home cooling that contribute to comfort and health. Several features should be noted:

- The different types of terminology used for nodes at each level
- The means-ends relations (two-headed arrows between levels) (see Box 1 for a discussion of interdependencies between functional areas)
- The reason for a node at one level is shown by its connection to one or more nodes at the next highest level
- The structural means of satisfying a functional requirement are shown by the means-ends links to nodes at the next lowest level
- Decompositions are shown by dashed, single-headed arrows within levels
- A means-ends link between two functional areas to depict interdependency (see Box 1); in this case, the interdependency has a disruptive influence and so the means-ends link is depicted by a dashed line.

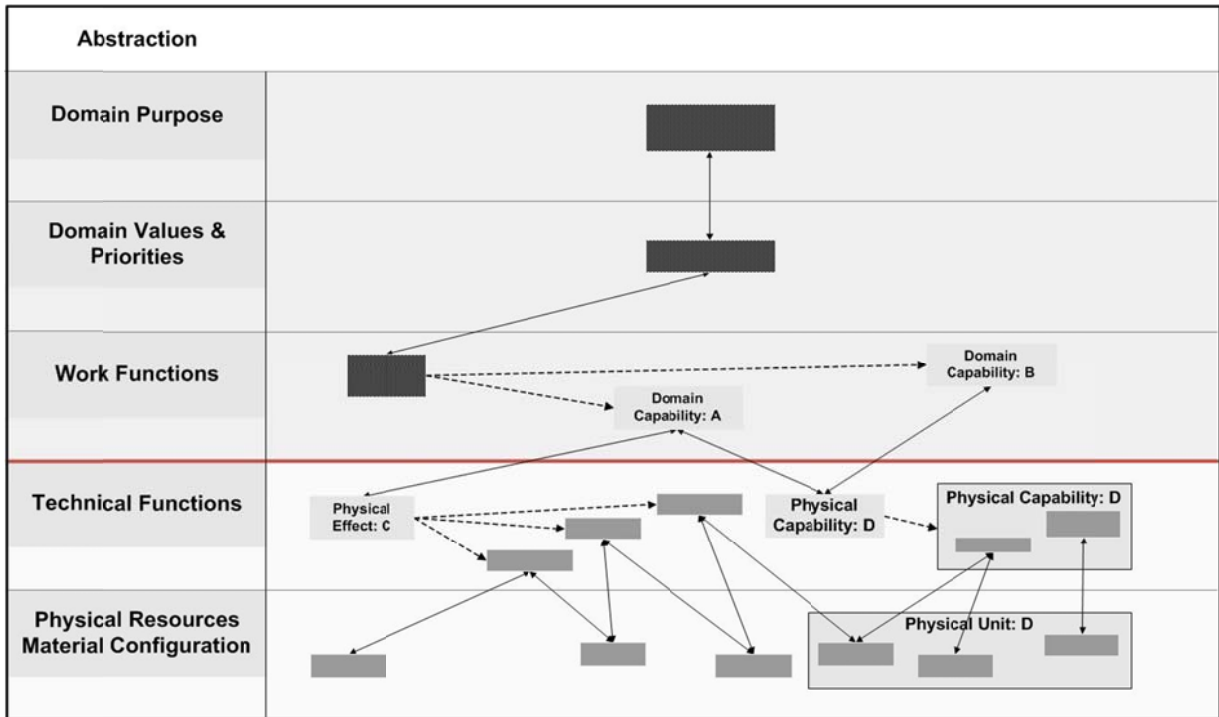


Figure 4: An alternate format for an abstraction-decomposition space

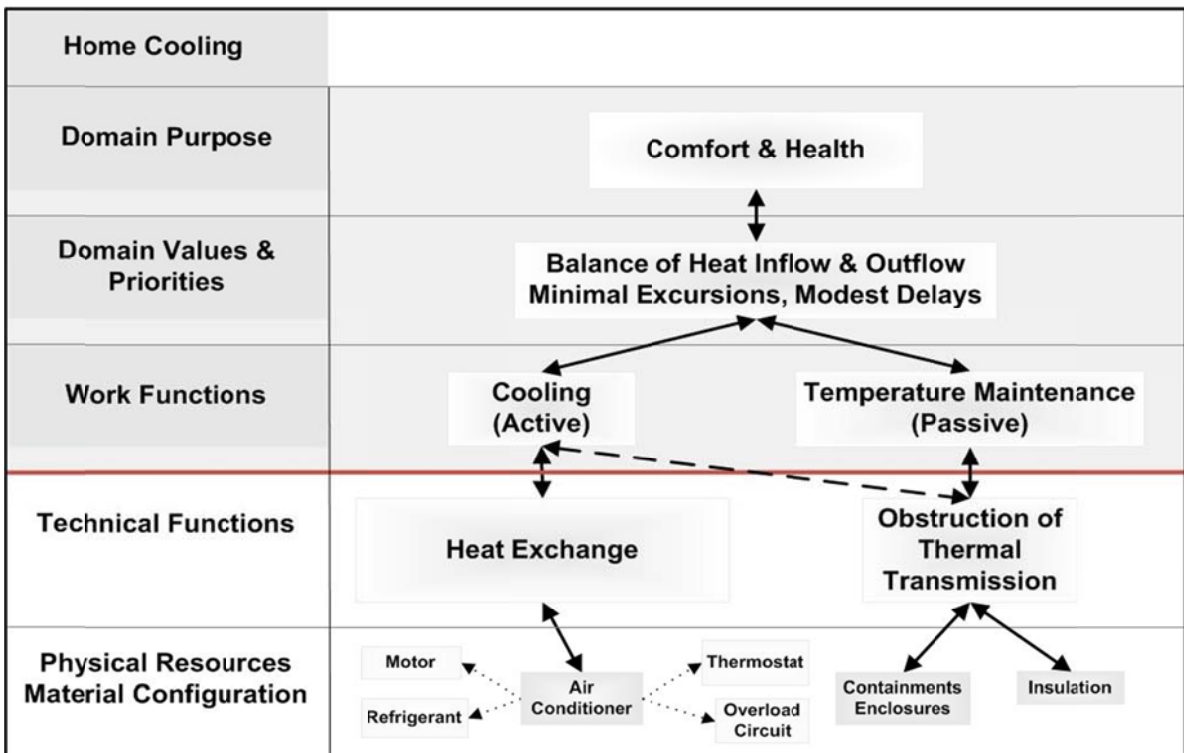


Figure 5: This abstraction-decomposition space shows some of the functional elements for home cooling that contribute to comfort & health

### **Box 1: Functional Interdependency**

Many systems have subtle interdependencies between functional areas; interdependencies that can be neglected during analysis. As illustrated in Figure 5, an abstraction-decomposition space maps interdependencies explicitly. Figure 5 shows both active and passive systems for temperature management but also shows interdependency between them. In this case, means-ends links from passive objects (insulation, enclosures) to the technical function for the active sub-system shows that the properties of the passive systems can impact the effectiveness of the active system. The illustration is drawn from an actual situation in which a home cooling system's overload circuit tripped on hot days because of defective insulation.

In Figure 5, the means-ends link from the passive to the active leg indicates that effective insulation supports the circulation of cool air (indirectly by reducing the possibility of system overload). Conversely, poor insulation compromises the circulation of cool air by increasing the possibility of system overload.

If the cooling system fails on hot days, should we replace it with a higher capacity cooling system or should we install better insulation? Work domain analysis does not answer that question but it does represent the options in a manner that helps you understand what you might do.

Note that the abstraction-decomposition space is not a causal map. We might come to understand via some form of causal, functional flow or process analysis, that defective insulation can cause the overload circuit to trip but that is not the role of work domain analysis.

## **An Abstraction-Decomposition Space is a Design Artifact**

An abstraction-decomposition space is not a design or even a design specification but rather a design artifact. It organizes information in a systematic manner that will support design. For example, it can be used to specify the information requirements of a work domain. Each node in the abstraction-decomposition space points to information (either directly or indirectly) that must be provided within the workspace, although different stakeholders (staff members, operators) will need access to different constellations of that information. This information will reveal to the workers the essential functional properties (purposes, values, resources and opportunities) of their work area. Considerable effort remains beyond the development of the abstraction-decomposition space to generate the required design specifications.

## **An Emphasis on Socio-Technical Systems**

Many who undertake work domain analysis focus on the technical aspects of the systems they analyze but the central concern here is with socio-technical systems and so the value of an analysis is limited to the extent that we do not consider the social aspects of the system. Figure 6 contrasts a predominantly technical with a predominantly socio-technical analysis. An instruction from a procedures manual for librarians was used to develop the abstraction hierarchy in the right panel:

*When a book is returned, draw a line with a black Magic marker through the name to protect the privacy of the borrower, replace the card in the book and then return the book to the shelf.*

That abstraction hierarchy was developed as a tutorial exercise in making the design rationale for this instruction explicit. Why a magic marker? Why black? These become apparent in relation to the privacy issue but other types of markers may do as well. Or are there additional reasons? The general

issue here is that design rationale is generally not made explicit in procedural specifications and workers can adapt without realizing they are violating the design rationale. The development of an abstraction-decomposition space is a step towards developing an explicit representation of design rationale that can be made available to librarians so that they can respond not necessarily to the technical meaning of this instruction but to the intent behind it.

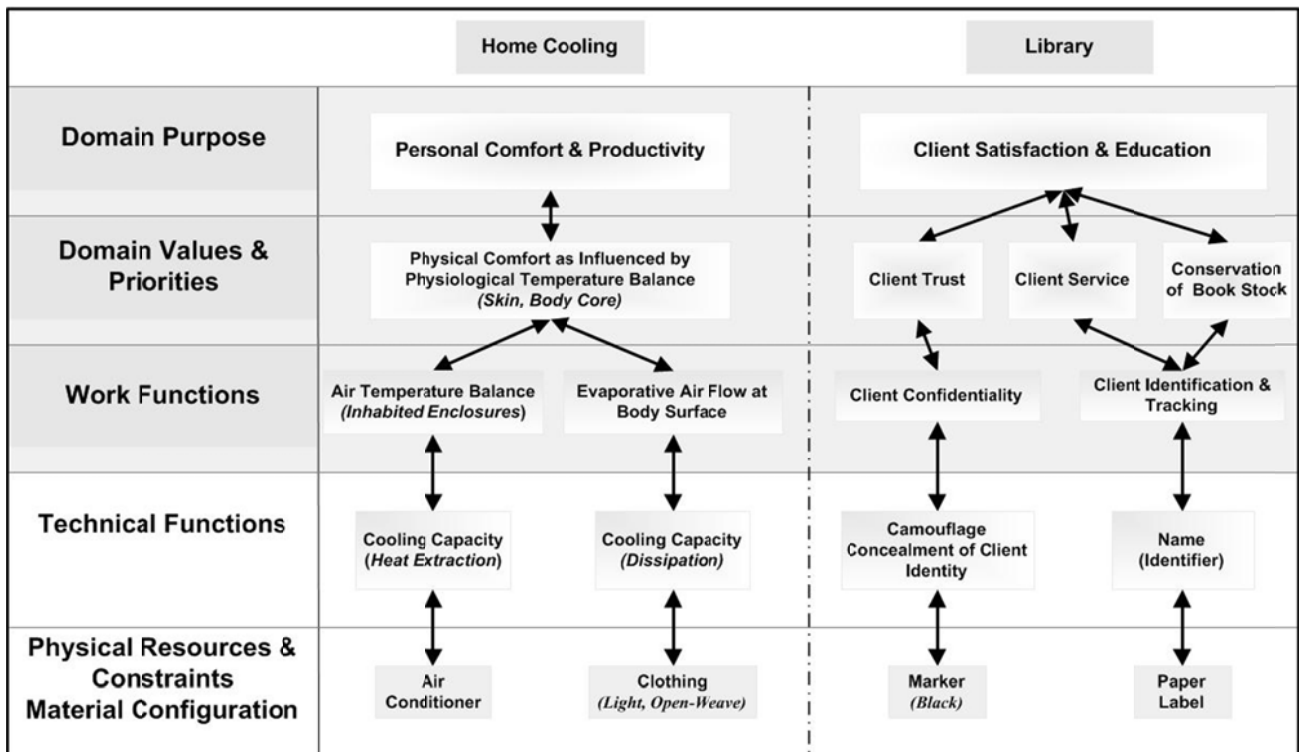


Figure 6: Two tutorial examples of the abstraction-decomposition space, a primarily technical system (Home Cooling, left) & a socio-technical system (Library Client Tracking, right)

## Socio-Technical Systems have Strange Properties

The functional properties of engineered systems are designed to be stationary. Non-stationarity intrudes when parts wear out but in the main, things stay the same. Such is not the case when we insert humans into the system. Humans change in themselves (they learn, they develop, they age) and they frequently modify the functionality of the systems they use. There is added value in laying out those functional properties that are non-stationary because of human participation. Unlike the non-stationarity of technical systems, human-induced non-stationarity will often enhance system effectiveness and should be promoted rather than avoided.

The additional contrast of a predominantly technical versus socio-technical analysis as shown in Figure 7 and Figure 8 was developed to illustrate this point. The IPOD example is characteristic of many of the analyses found in the literature but the theatrical example illustrates a challenging feature of socio-technical systems that is often neglected. The functionality of the parts can change as the system evolves. The director might take on acting functions and actors might take on directing responsibilities. In addition, actors might develop and adapt their capabilities. For example, a comedy specialist might develop as a dramatic actor and, over the life of an extended production, might begin to inject dramatic elements into the performance. How does that change the system; in this case the theatrical experience for audience, performers and director?

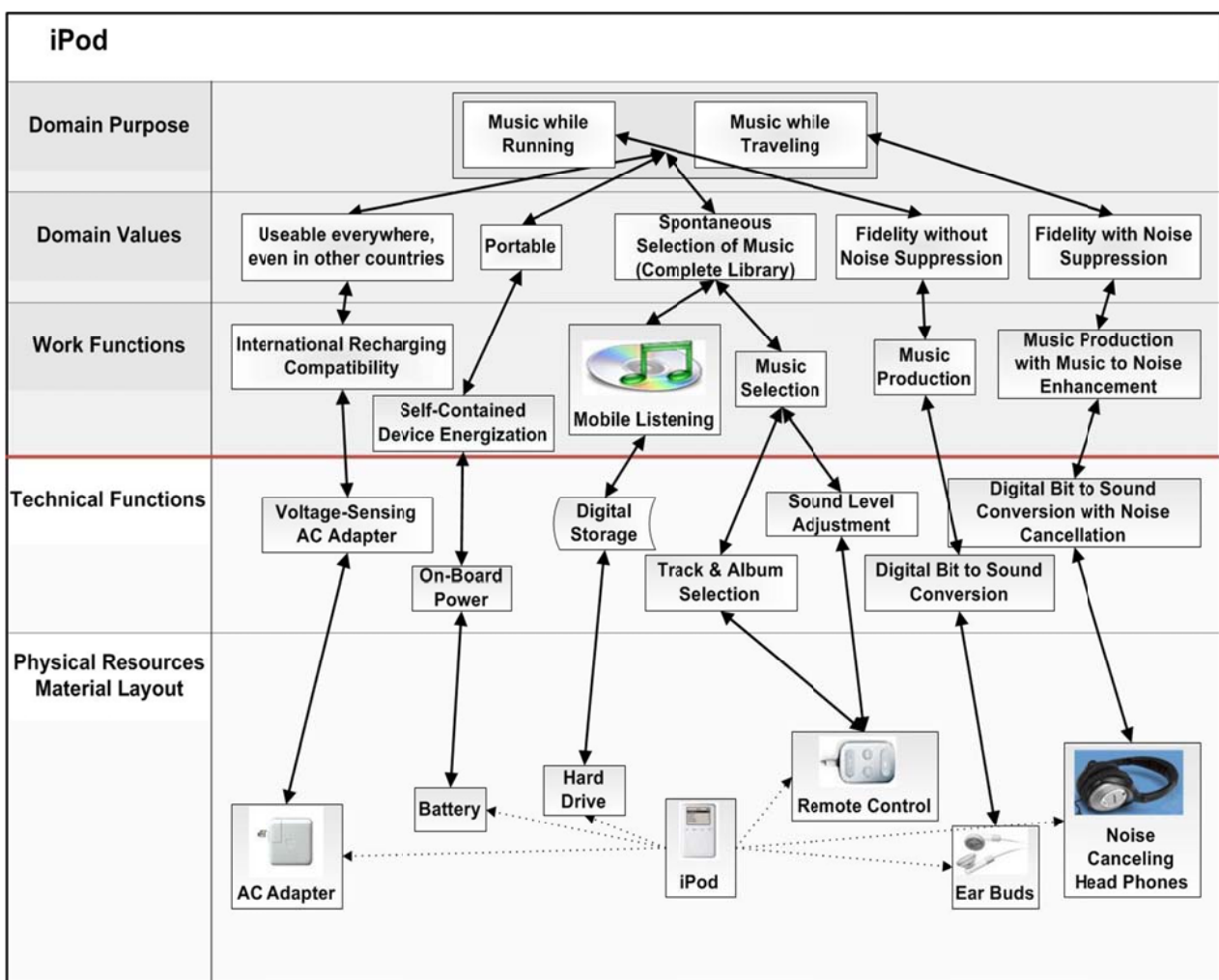


Figure 7: A predominantly technical analysis of an IPOD

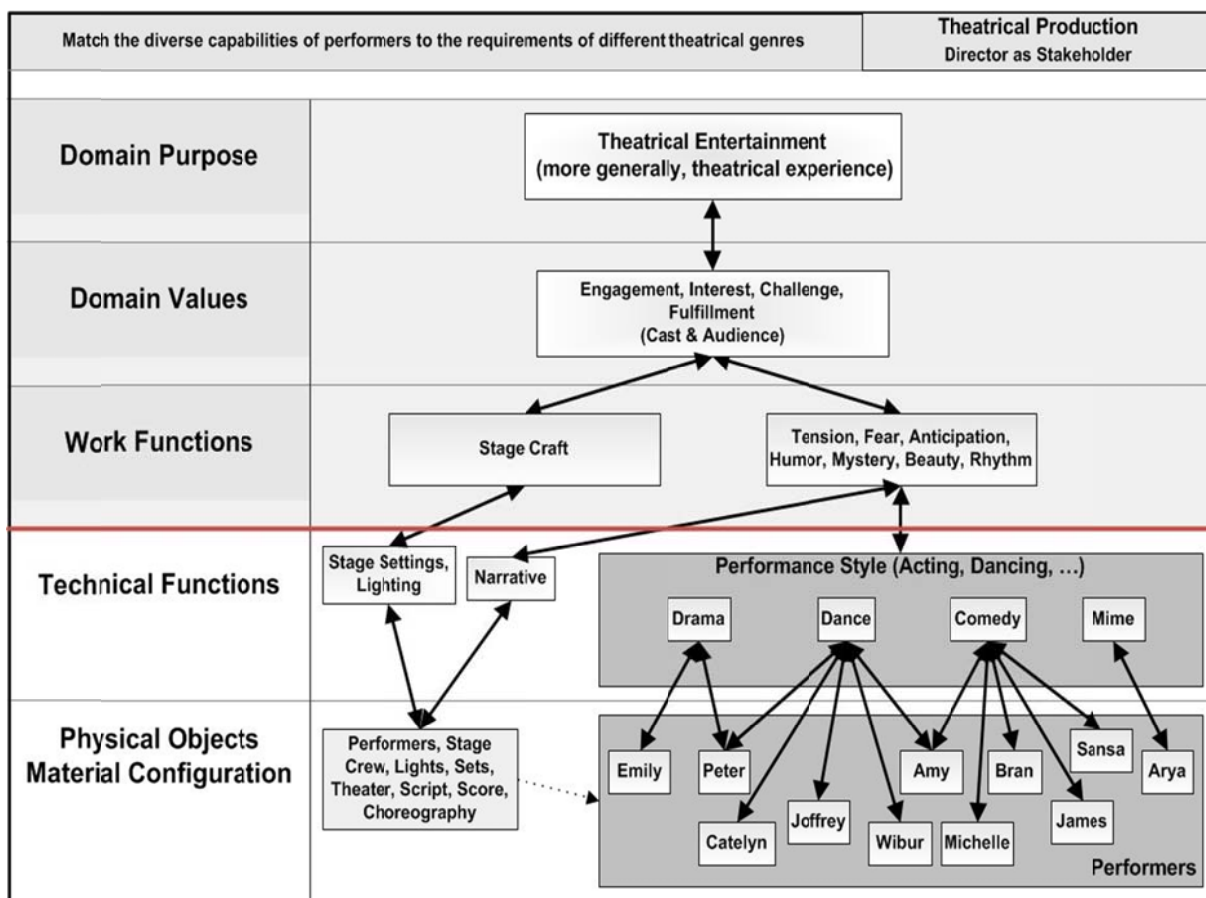


Figure 8: A socio-technical analysis of a theatrical production

## Why Abstraction-Decomposition?

Why might we believe that an abstraction-decomposition space is a useful form of representation? Jens Rasmussen has shown that expert trouble-shooters and expert problem-solvers navigate through an abstraction-decomposition space as they solve problems. Typically, they start with purposes or values at the system level and then work down towards decompositions at physical object and technical function levels. Also, typically, the trajectory is irregular, opportunistic and iterative.

This pattern is depicted in Figure 9 and Figure 10 (adapted from Hoffman and Lintern, 2006) where a fragment of a work domain for weather forecasters (Figure 9) is overlaid with a scenario trajectory (Figure 10). The numbers trace the sequence in which one subject matter expert visited the nodes of the abstraction-decomposition space within this scenario. The callouts reveal the comments made by the subject matter expert at each node.

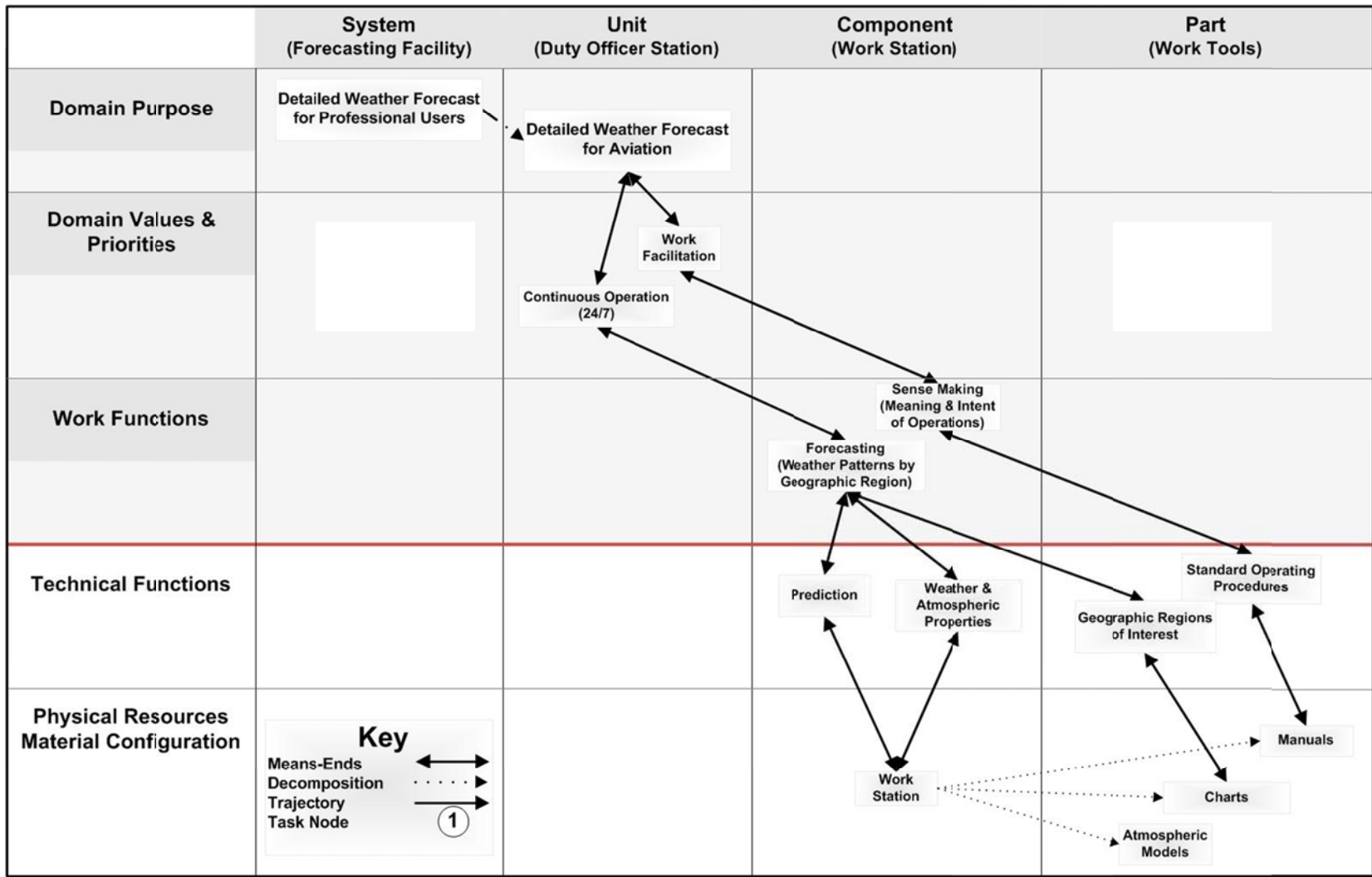


Figure 9: An Abstraction-Decomposition space for a fragment of a weather forecasting work domain

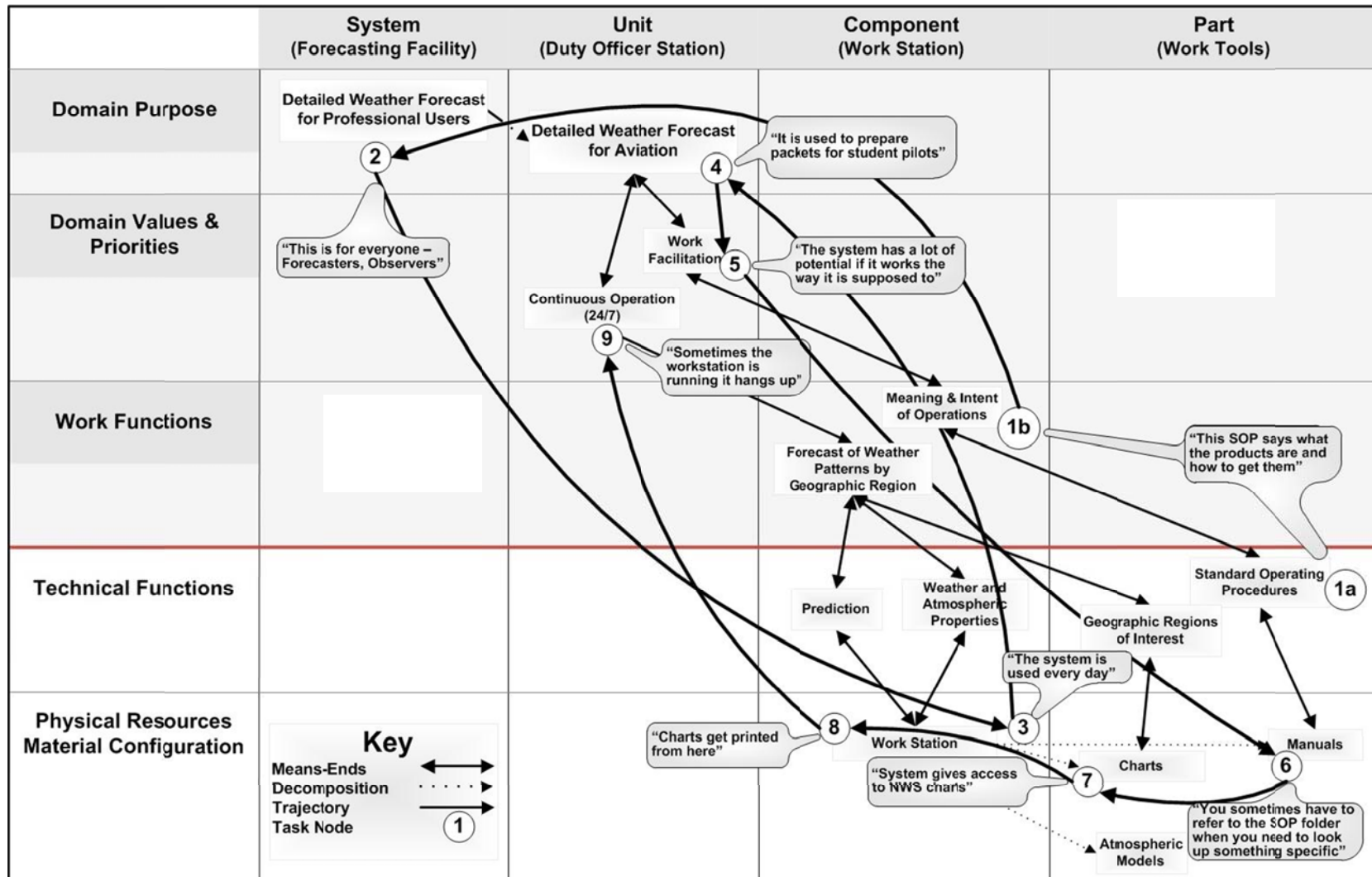


Figure 10: Figure 9 overlaid with a scenario trajectory described by one subject matter expert for resolution of a weather forecasting work problem

The claim is that we all do that (at least implicitly) every time we solve a moderately complex problem. The commitment to the abstraction-decomposition space constitutes a theory of cognition, albeit one that has not yet been enunciated explicitly or in detail. Can you believe the claim that we navigate through an abstraction-decomposition space when we plan or solve problems? If you cannot, you probably should not build abstraction-decomposition spaces. On the other hand, if you can believe that this is fundamental to the way people behave, you should find work domain analysis to be a useful exercise. Furthermore, your understanding of that theory is your best guide to how you construct the abstraction-decomposition space; what the levels of abstraction mean, what sort of concepts go into each of those levels, and how you deal with decomposition.

Also remember that the abstraction-decomposition space is not a formal system as is, for example, mathematics. The elements of a formal system must be defined precisely and the relationships between them must be logical and consistent. Many of the critiques in the literature point to logical inconsistencies in the way that the various relationships of the abstraction-decomposition space are defined and used but remember that this was never intended to be a formal system. It was always intended to be a depiction of the way subject matter experts could think effectively about their work domain. Many appear to believe that thinking processes correspond to formal computational processes but the underlying assumption of work domain analysis is that the most effective forms of thinking are irregular, opportunistic and iterative; they are not irrational but they are very far from anything like a formal computational process.

## The Relationship to Systems Engineering

Some Systems Engineers dismiss the contribution of a work domain analysis. They claim that this is a Systems Engineering tool that is already discussed adequately in standard textbooks. However, other Systems Engineers state that nothing like work domain analysis exists within their discipline.

Decomposition is used extensively, systematically and explicitly within Systems Engineering (e.g., Blanchard and Fabrycky, 2006). In contrast, the commitment to functional abstraction is less clear. Tools such as Attributes Lists (which organize system features into the three broad categories of objectives, constraints and functionality), Hierarchical Objective Lists (which can be configured into Objective Trees) and Morphological Charts (also referred to as Function-Means Charts and Concept Combination Tables) are activity-independent analyses that use dimensions of classification somewhat like the abstraction dimension of work domain analysis. Nevertheless, it is not clear that these tools are used widely or that their products are well integrated into the Systems Engineering process. In addition, although Systems Engineering texts refer to purposes and values, they do not clarify how those purposes and values should be integrated into the design of a system or how they might be implemented as constraints on design and use.

The magnitude of the intellectual debt owed Systems Engineering by cognitive work analysis remains unclear, but of more concern is whether the products of cognitive work analysis can be of value within the broad scope of the systems design process. The design of complex socio-technical systems continues to pose significant challenges, one of which is the effective deployment and use of human resources. Anything we can do as cognitive engineers to resolve those challenges will be of considerable benefit and the fact that the major concepts of work domain analysis are already familiar to Systems Engineers is a point of contact between the two disciplines that should support rather than detract from this effort.

## Source Material

Blanchard, B. S., & Fabrycky, W. J. (2006). *Systems Engineering and Analysis* (4<sup>th</sup> ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Hoffman, Robert R. & Lintern, Gavan (2006). Eliciting and Representing the Knowledge of Experts. In K. Anders Erin, Neil Charness, Paul J. Feltovich & Robert R. Hoffman, *The Cambridge Handbook Of Expertise And Expert Performance*. New York: Cambridge University Press, Ch 12, pp 203-222.

Rasmussen, J., Petjersen, A. M., & Goodstein, L. P. (1994). *Cognitive systems engineering*. New York: Wiley.

Vicente, KH (1999). *Cognitive Work Analysis: Towards safe, productive, and healthy computer-based work*. Mahwah, NJ: Lawrence Erlbaum Associates.

# Appendix A: Abstraction Levels

---

## Domain Purpose

What do we want to achieve with this system?

Why does the system exist? Why is it being designed?

A purpose may be multi-dimensional.

Note the distinction between goals and purposes. Goals are states to be achieved, or maintained, by actors at particular times. They are attributes of actors and they are dynamic. Purposes are the overarching intentions a work domain was designed to achieve. Purposes are properties of domains, not of actors, and that they are relatively stationary.

## Domain Values & Priorities

What are the values that shape how we use this system, specifically how we use it to satisfy the purpose? What abstract properties help us establish priorities with respect to domain purpose? What are our guiding concerns?

What considerations guide what we do? Most significantly, what considerations constrain how we set priorities and allocate resources?

Properties of balance, conservation, preservation, minimization and maximization are important, e.g. safety-productivity trade-off.

Policies and Legislation will shape strategies, e.g. Rules of Engagement, Geneva Convention (what is the underlying value?)

## Work Functions

These are the essential device-independent functions irrespective of physical nature of the system, e.g., communication with reference to domain purpose via values and priorities.

## Technical Functions

These are the specific functions of the physical elements of the system; the properties necessary and sufficient for control of physical work activities and use of equipment.

## Physical Resources & Material Configuration

Specify names of physical devices, colors, shapes, locations, etc. But only introduce descriptors that are relevant to your design purpose. For example, shape of an aircraft is not useful if you are designing a flight simulator for that aircraft but is useful if you are designing an aircraft identification system.

Properties necessary and sufficient for classification, identification and recognition of particular material objects and their configuration, and for navigation through the system.