1. Introduction

This technical report’s purpose is to review, judge, and recommend a modeling abstraction technique to assist developers with understanding and designing ground-breaking Human Robotic Interface systems (HRI systems) for the Chemical, Biological, Nuclear, Radioactive, and Explosive device (CBNRE) domain. The CBRNE domain is characterized by multiple decision sequences, timing, and hierarchical relationships. Capturing these constraints is paramount to understanding and modeling the team decision-making process present in the CBNRE domain [Gonzalez 2004]. A modeling technique designed to represent these constraints will facilitate understanding the decision-making process, providing how and where to incorporate HRI systems to augment and improve the CBNRE response time and quality.

An example from the CBNRE domain is used throughout this paper to facilitate the understanding of the differences between the reviewed techniques. The example is based on the CBNRE domain task of Incident and Hazard Mitigation. The Incident and Hazard Mitigation task involves three focuses: isolating the contaminant or hazard, mitigating the hazard’s effects, and establishing and securing a perimeter around the hot zone (or the area in which the hazard’s effects are the most dangerous). Throughout the review discussion, particular focus will be given to the Construct Perimeter sub-task to illustrate the differences between each technique. Modeling the Incident and Hazard Mitigation task requires the abstraction techniques to express the interconnectivity of the various subparts, express partial ordering of these subparts, and be tractable. Tractability in this context means that the modeling technique can model the CBNRE domain via reasonable time and resources.

The reviewed modeling abstraction techniques can be parsed into three groups according to their basic modeling characteristics. The modeling characteristic denotes whether the abstraction is primarily designed to model functions and goals or data. An abstraction that is designed to model functions and goals (henceforth referred to as a functional abstraction) will express the model in terms of goals and the functions or processes required to complete those goals. The information, or data, needed to complete a function may or may not be explicitly represented in a functional abstraction. The opposite of the functional abstraction is an abstraction designed to model data (henceforth referred to as a data abstraction) that expresses the modeling in terms of the path or flow of the information or data.
Functions will be represented in a data abstraction but only in terms of how the functions consume, alter, or create data. Goals are usually not explicitly represented in a data abstraction. The third group of modeling abstractions techniques reviewed is termed crossover abstraction. A crossover abstraction is one that is either primarily a functional or data abstraction that in some limited way crosses over and represents elements from the other abstraction group, e.g. a functional abstraction that explicitly represents information needed by each function. The abstraction techniques reviewed are presented in their perspective groups: functional abstractions, data abstractions, and crossovers abstractions.

2. Functional Abstractions

Functional abstractions have been used in a large number of applications. There are three functional abstraction techniques that have seen widespread use in modeling human centric workflow, making them relevant to modeling the CBNRE domain. The three techniques are the Hierarchical Task Analysis (HTA) [Shepherd 1998], Work Domain Analysis (WDA) [Vicente 1999], and the Constraint-based Task Analysis (CbTA) [Vicente 1999]. The latter two models, WDA and CbTA, are part of Cognitive Work Analysis (CWA) [Vicente 1999]. Two other models Goal-Directed Task Analysis (GDTA) [Endsley et al. 2003] and Sensor-Annotated Abstraction Hierarchy [Reising and Sanderson 2002a, b, and c] are also functional abstractions; however, these models have crossover features and are therefore discussed in Section 4.0. Functional abstractions are designed to represent the functions or actions that comprise a system in a hierarchical fashion. The relationship between the functions and the ordering of functions is what differentiates the functional abstraction techniques.

2.1. Hierarchical Task Analysis

The Hierarchical Task Analysis (HTA) technique typifies functional abstractions and has a long history with many variations, extensions, and simplifications [Shepherd 1998]. The term encompasses ideas developed by Annett and Duncan in the late 1960’s and early 1970s [Annett and Duncan 1967; Annett et al. 1971; Duncan 1972; and Duncan 1974]. The concept of HTA is to define tasks via a hierarchy of goals and plans composed of subordinate goals and plans. Often goals at higher levels are more abstract or general and goals at lower levels better resemble tasks or functional steps. However, the actual definitions of these nodes and indeed the word “task” itself are somewhat fluid and have been under considerable debate [Shepherd 1998].

Essentially HTA is a directed graph with a root node and subsequent child nodes linked together generally by a part-whole relationship. These nodes sometimes represent goals, tasks, plans, and behaviors [Shepherd 1998]. Regardless of how the nodes are defined, they represent a function that must be completed to achieve the objective of the parent node. The sheer flexibility of HTA and its focus on functional understanding for the entire domain makes it applicable to the CBNRE domain and a valid modeling technique. Its focus on functions makes it easy to understand and communicate to subject
matter experts. However, HTA provides no inherent mechanisms for scheduling, representing parallelism, or information, all of which are vitally important to the CBNRE domain.

The Incident and Hazard Mitigation task has been modeled in Figure 1 using a HTA. Notice how clean and consequently simple the model is. Its flexibility is represented in the nodes’ adaptability to represent any functional concept (task, goal, sub-goal, etc.). The HTA is simple because it does not model any notion of the information used by the system in decision making or any scheduling required by the system. The Construct Perimeter sub-task is represented as a child node of Establish Perimeter, meaning that it is part of the Establish Perimeter node and because of HTA’s simplicity, no further information is provided about the Construct Perimeter sub-task. However, the successful completion of the task and development of design considerations requires a modeling technique that can represent additional sub-task elements such as scheduling and timing, information required for decision making, and what decisions are made as part of the sub-task. In summary, HTA is applicable to the CBNRE domain, has limited expressiveness, has limited ordering representation, and is tractable.

Figure 1: The Incident and Hazard Mitigation task modeled via Hierarchical Task Analysis.

2.2. Work Domain Analysis
Cognitive Work Analysis has many components, one of which is the Work Domain Analysis (WDA) [Vicente 2001]. The WDA has a similar scope as the HTA, the entire domain, and its purpose is to model
the constraints of the work domain to create a detailed understanding of the domain’s modeling. The model technique used to perform a WDA has traditionally been an abstraction hierarchy performed in an abstraction-decomposition space, also collectively referred to as an abstraction-decomposition [Rasmussen 1986]. The abstraction hierarchy was developed and formalized by Rasmussen [Rasmussen 1979, 1986, 1988; and Morey et al. 1994]. The abstraction-decomposition has been used by many individuals [e.g. Cummings 2004; Krosner et al 1989; and Gersh 2005].

The abstraction hierarchy is a modeling language similar to HTA; however, it has two dimensions to the hierarchy representing different relationships and specified levels of abstraction. The two hierarchies are: a means-end relationship along the vertical axis and a part-whole relationship along the horizontal axis. The horizontal axis, and therefore the horizontal hierarchy, is in essence a HTA. Where the abstraction hierarchy differs and possibly improves upon a HTA is in its vertical hierarchy, that of a means-end relationship. The standard five levels that comprise the vertical hierarchy are as follows: functional purpose, abstract function, generalized functions, physical functions, and physical form [Rasmussen 1986; Lind 1999]. The HTA and the abstraction hierarchy have been compared and the abstraction hierarchy was concluded to provide deeper knowledge and a fuller set of system constraints and capabilities, while the HTA was assessed to be a more procedural, human-centered approach that is easily learned and applied [Burns and Vicente 2001]. The extra expressiveness of the abstraction hierarchy fundamentally comes at the cost of human readability. This readability may become an issue in the CBNRE domain with regard to interactions with subject matter experts and designers unfamiliar with the abstraction hierarchy’s double hierarchy. The abstraction hierarchy, as with the HTA, provides no inherent mechanisms for scheduling or representing parallelism.

The Incident and Hazard Mitigation task has been modeled in Figure 2 using abstraction-decomposition. Unlike the HTA in Figure 1, the model in Figure 2 includes objects, which are in the bottom right of Figure 2. Objects represent both tools and resources. The tools implicitly represent some of the system’s information sources; however, they do not represent all of the information available, nor do they represent information that is gathered or transformed by methods not represented in this technique. This lack of information leaves the designer uncertain as to what information is essential, what information is not optimal for decision making, and what information can safely be altered or improved, which is our goal in working in the CBNRE domain. The Construct Perimeter sub-task is, as it was in HTA, a child node of Establish Perimeter; however, in this technique, the Construct Perimeter represents a means to the end of Establish Perimeter and not a part of Establish Perimeter. This change in child node meaning implies correctly that a perimeter can be established without a physical perimeter being constructed. The Construct Perimeter sub-task in the abstraction hierarchy has its own children nodes representing objects used as means to complete the Construct Perimeter task. These two changes
demonstrate the abstraction hierarchy’s expressiveness and applicability beyond the HTA. The abstraction hierarchy suffers from the same limitations as HTA. Specifically, the abstraction hierarchy does not permit the representation of scheduling and timing, the information required for decision making, and what decisions are made as part of the sub-task. In summary, the abstraction hierarchy is applicable to the CBNRE domain, is somewhat expressive, has very limited ordering representation, and is tractable.
### Incident & Hazard Mitigation System

<table>
<thead>
<tr>
<th>Subsystems</th>
<th>Function Units</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract Function</td>
<td>Contain and control the incident/hazard</td>
<td>Stop Hazard, Contain Hot Zone</td>
</tr>
<tr>
<td>General Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objects</td>
<td>Decontamination Equipment (not for people, for people see victim care)</td>
<td>Construction Resources, Monitoring equipment, Detection kits</td>
</tr>
</tbody>
</table>

**Figure 2:** The Incident and Hazard Mitigation task modeled via Work Domain Analysis’ Abstraction-Decomposition.
2.3. Constraint-based Task Analysis

The Constraint-based Task Analysis (CbTA) models functions, or tasks in the CbTA terminology, in a way that depicts their relationships between functions as action-means-end relationships [Vicente 1999]. It is this relationship that forms the foundation of the CbTA. The CbTA model provides some mechanisms for the scheduling of functions because of its inherent relationship type and the modeling techniques it traditionally employs, unlike the HTA or the abstraction hierarchy. As with all functional abstractions, data or information is not explicitly represented, only the functions and their resulting goal states are represented. There have been several modeling languages employed to express the CbTA model. The most common modeling language is Decision Ladders [Rasmussen 1986].

There are a number of issues with respect to modeling expressiveness with decision ladders that have caused others to modify or replace it as the CbTA modeling language [e.g. Jones et al 1993]. Decision ladders, unlike the dataflow languages discussed in Section 3, are inherently awkward at expressing parallelism or complex partial order scheduling [Jones et. al 1993]. This awkwardness is a result of decision ladders being fundamentally based on finite state machines. However, when a decision ladder involves more than one decision sequence or the decisions overlap in time, the finite state machine model is inadequate, as it cannot represent parallelism succinctly [Johnston 2004]. Jones et al. [1993] extended decision ladders for use with two parallel operators; however, this is still inadequate for the CBNRE domain with hundreds if not thousands of potential actors.

Humphrey et al. have substituted statecharts [Harel 1987] for decision ladders; however, this substitution is unusual and may not be generally accepted [Humphrey et al. 2006]. Statecharts permit succinct, concurrent decisions through the addition of special groups that represent families of functions that can be simultaneously performed. Statecharts are designed for representing individual tasks or function groups as were decision ladders, as required by CbTA. The CbTA’s limited scope and its traditional reliance on decision ladders prevent it from being a true candidate for modeling the CBNRE domain.

Part of the Incident and Hazard Mitigation task has been modeled in Figure 3 and 4 using the decision ladders technique. Figure 5 models the entire Incident and Hazard Mitigation task using the statecharts technique. Notice that the techniques depict order; however, the statecharts technique represents parallelism and congruence clearer than the decision ladders technique. This ordering is not represented in the HTA or the abstraction hierarchy techniques mentioned in Sections 2.1 and 2.2, respectfully. Furthermore, the CbTA has a two step structure: an information processing activity node followed by a state of knowledge node. The two step structure hints at information processing but, like the abstraction hierarchy technique, does not explicitly represent the information being used at each node. Therefore these techniques suffer from the same deficiencies as the abstraction hierarchy technique with regards to
not representing all of the available information and not representing what elements can be improved or altered by adding new systems, such as HRI systems in the case of the CBNRE domain.

The decision ladder version of the CbTA in Figure 3 focuses only on the Establish Perimeter sub-task of Incident and Hazard Mitigation. A second decision ladder was developed to model the isolation and mitigation of the hazard, see Figure 4. This separation is necessary because these two tasks occur in parallel and decision ladders cannot represent parallelism clearly without separating each parallel task into its own model. The decision ladder version of the CbTA depicts the Construct Perimeter sub-task as four nodes in a loop (see bottom right of Figure 3). These loops, in decision ladders, represent that a task can be ongoing. The two-step structure hints that certain information (namely event type, scale, possible escalation, methods, and available materials) is employed when determining how to complete the construction of a perimeter.

The statechart version of the CbTA, in Figure 5, depicts the entire Incident and Hazard Mitigation task because of its ability to represent parallelism clearly. This is in contrast to decision ladders. The same four nodes that represented the Construct Perimeter sub-task in the decision ladder version are also present in the statechart version. However, because of parallelism, more parent nodes are represented in the statechart, providing a richer perspective of what events and thoughts lead to the construction of a perimeter.

The two step structure of the CbTA (present in both Decision Ladders and statecharts) increases the expressiveness of the modeling technique by representing information process, although in a somewhat limited perspective. However, the CbTA does not represent the information used to make decisions and does not clearly represent the decisions to be made as part of the sub-task. In summary, the CbTA with either method representation is somewhat applicable to the CBNRE domain, is expressive, represents ordering very well via statecharts and decently using decision ladders, and is tractable.
Figure 3: The Incident and Hazard Mitigation task modeled via Constraint-based Task Analysis using the Decision ladder technique, Part 1.
Figure 4: The Incident and Hazard Mitigation task modeled via Constraint-based Task Analysis using the Decision ladder technique, Part 2.
3. Data Abstractions

3.1. History and Overview

Data abstractions have been used primarily in software, signal processing, and embedded system designs [Johnston 2004]. The primary type of data abstractions have been the family of dataflow programming languages. Dataflow languages were developed in response to the belief that von Neumann processors and their corresponding languages were inherently unsuitable for the deployment of parallelism [Dennis and Misunas 1975]. Dataflow was designed to embrace parallelism by focusing on the data and executing instructions as soon as their local data was available. Therefore, dataflow only imposes a partial ordering constraint on execution, thereby providing for parallelism to be exploited.
Dataflow system resembles a directed graph with the data flowing between instructions (the graph nodes) according to the arcs in the graph [Dennis and Misunas 1975].

Dataflow languages have, since the 1990’s, become more visual in nature and are called visual dataflow programming languages [Johnston 2004]. The visual dataflow programming languages have been refined and developed by a number of individuals [Auguston and Delgado 1997; Baroth and Hartsough 1995; Bernini and Mosconi 1994; Ghittori et al. 1998; Green and Petre 1996; Harvey and Morris 1993, 1996; Hils 1992; Iwata and Terada 1995; Morrison 1994; Mosconi and Porta 2000; Serot et al. 1995; Shizuki et al. 2000; Shürr 1997; Whiting and Pascoe 1994; and Whitley 1997]. The two most prevalent commercial dataflow visual programming languages are MathWorks’ Simulink and National Instruments’ LabVIEW. During the development of dataflow visual programming languages the focus slowly shifted from exploiting parallelism to the advantages that data abstractions provide to the developer during the software development lifecycle [Johnston 2004, Baroth and Hartsough 1995]. Baroth and Hartsough reported that developing systems in a visual dataflow programming language, namely LabVIEW, was considerably faster, four to ten times faster, than procedural functional languages such as C [Baroth and Hartsough 1995]. They attributed the speed improvement to dataflow’s ability to explicitly and visually show the information processing.

3.2. Pure Dataflow Model

The pure dataflow model is a directed graph with the nodes representing instructions and arcs representing the data dependencies between instructions [Johnston 2004, Dennis 1974]. The data flows on the arcs conceptually as data tokens or packages and will queue before an instruction in an unbounded first-in, first-out (FIFO) queue [Kahn 1974]. Figure 6 depicts how a dataflow abstraction would represent Incident and Hazard Mitigation task. The figure shows the processing nodes as rounded cornered rectangles and the information being sent along the arcs as traditional rectangles. The dataflow technique, being a data abstraction, explicably represents information used at each step and produced at each step in contrast to the previous functional techniques discussed in Section 2.

The dataflow technique still has a hierarchical nature that is similar to the HTA; however, the dataflow hierarchy is determined by the flow of information and not the decomposition of goals or tasks. This shift in hierarchy meaning caused the Construct Perimeter task that was a sister node to Maintain Site Security in the HTA technique (Figure 1) to become a child of Maintain Site Security in the dataflow technique (Figure 6). This change occurred because the perimeter information is generated from that task of constructing the perimeter and is necessary in evaluating and maintaining the security of the site. The Construct Perimeter task has children nodes in the dataflow technique that represent information pieces that are used in fulfilling the task. These children nodes are a key distinction between the dataflow technique and previous functional techniques, as these nodes provide a richer representation of the
Construct Perimeter task information needs. This richer representation provides more information pieces than the knowledge state depicted in the two step task-knowledge structure of the CbTA leading into the Construct Perimeter task. However, the dataflow technique does not represent scheduling or clearly what decisions are made as part of the sub-task. In summary, general dataflow is very applicable to the CBNRE domain, is quite expressive, represents partial sub-task ordering, and is tractable.
3.3. Dataflow’s expressiveness and driving factor

Over time the expressiveness of the dataflow language has increased so that any arbitrary system can be represented in a dataflow abstraction [Johnston 2004]. Much of the work to date has been related to
implementing a dataflow language on hardware or maximizing parallelism. Neither of these areas is of much interest to this paper’s aforementioned goal of evaluating modeling abstractions for the CBNRE domain, as the abstraction is to be a guide for development and will not be directly implemented. However, there are a number of papers and ideas that have increased or addressed aspects of dataflow’s modeling expressiveness that will be addressed in the remainder of sections.

The dataflow languages execute in one of two approaches. The first approach is data driven where execution of an instruction node occurs the moment all its data is present on its incoming arcs. This approach is termed the data-driven approach [Dennis 1974] or the data-availability-driven approach [Johnston 2004]. Execution requires two steps: wait passively until all required incoming data is present, and then processes the data tokens placing the output data tokens on all appropriate outgoing data arcs [Johnston 2004, Dennis 1974].

The second approach is also data driven; however, execution of an instruction node occurs after its output data is requested and all relevant input data is present. This approach is termed the demand-driven approach [Kahn 1974]. Execution in this approach has four steps: an instruction node has its output information requested, the instruction node in turn requests output from all its relevant input arcs, the input arcs present all the requested data, and finally the instruction node consumes the data tokens placing the requested output data on all appropriate outgoing arcs [Davis and Keller 1982; Johnston 2004].

3.4. Scheduling dataflow

One issue that is not addressed in the pure dataflow model is execution scheduling. This is important in cases where there are limited resources to perform the executions or where the control of execution needs to be separated and possibly scheduled apart from its input data. The execution in a pure dataflow model always occurs when all relevant input arcs have data; however, there are systems that require the execution to be controlled. Essentially adding scheduling to the dataflow model adds a component of control to the system. The ability to increase the expressiveness of the dataflow technique to include scheduling is important to the CBNRE domain, as there are both limited resources and changing scheduling priorities as the events unfold that need to be captured in a modeling abstraction. However, care must be taken with regard to understanding scheduling in dataflow languages, particularly with languages termed dataflow synchronous languages or synchronous dataflow (SDF) [Benveniste 1994; Johnston 2004; Lee 1991]. Generally, the term synchronous dataflow is used to define a dataflow when the number of data tokens consumed and produced for a given instruction is fixed. This feature simplifies system scheduling because the system is deterministic. These synchronous dataflow systems do not necessarily have any scheduling control nor do they help manage limited resources.
Enabling scheduling and execution control in dataflow models requires the addition of two node types: the SWITCH and the SELECT nodes [Lee 1991]. Both of these nodes perform an if-then-else execution based on an input control signal. The node SWITCH determines which outgoing arc receives the incoming arc’s data. The node SELECT determines which incoming arc provides the data to the outgoing arc.

The Incident and Hazard Mitigation task example modeled with the Dataflow technique is provided in Figure 6 and is modified to illustrate the addition of one SWITCH statement in Figure 7. The SWITCH statement provides the ability to denote that the two processes, remove/eliminate hazard and decontamination, do not both necessarily occur in a CBNRE response and that the information regarding the object to be mitigated determines which process is chosen. This additional scheduling representation provides more modeling power than the basic dataflow method presented in Figure 6. However, these additional scheduling elements do not mitigate the dataflow’s inability to represent clearly what decisions are made as part of the sub-task. The Construct Perimeter task, being at the bottom of the dataflow model, does not change with the additional scheduling representation. In summary, dataflow with SWITCH and SELECT nodes is very applicable to the CBNRE domain, is quite expressive, represents ordering very well, and is tractable.
Figure 7: The Incident and Hazard Mitigation task modeled via the Dataflow technique with scheduling.

An extension to synchronous dataflow is the multidimensional synchronous dataflow (MDSDF) [Murthy and Lee 2002]. Multidimensional synchronous dataflow addresses the concern that the general dataflow arcs are modeled after first-in-first-out (FIFO) queues, which are inherently one-dimensional.
Multidimensional synchronous dataflow increases the dataflow expressiveness by generalizing the FIFO queues into arrays and introduces the concept of queue sampling windows [Murthy and Lee 2002]. A queue sampling window allows the decision node to determine its output based on the history of that type of input and not a single sample as was done in the original dataflow technique. This key change is important to the CBNRE domain modeling as most decisions will benefit from understanding the historical view of the information facilitating better quality decisions. This change will also increase the efficiency of computation given limited resources.

The Incident and Hazard Mitigation task is modeled using the multidimensional synchronous dataflow technique, as presented in Figure 8. The addition of rectangles on input arcs represents the sampling window or history information that is used in the decision making represented in the processing node. This history provides additional modeling power to further refine the nature of the system and separates information that is used instantaneously from information that is averaged before being used. An example of instantaneous usage is the information regarding secondary devices, which must be acted upon immediately. The information regarding secondary devices is produced from the node on the bottom, center of Figure 8 called “Search for secondary devices.” The Construct Perimeter task, in contrast, utilizes its required information in context with its history in order to understand trends and predictions, leading to better decisions regarding the construction of the perimeter. However, this dataflow technique does not represent clearly what decisions are made as part of the sub-task. In summary, multidimensional synchronous dataflow is very applicable to the CBNRE domain, is very expressive, represents ordering very well, and is tractable.
Figure 8: The Incident and Hazard Mitigation task modeled via the Multidimensional Synchronous Dataflow technique. The rectangles on the input arcs represent the multidimensional nature of that information (sampling window).
4. Crossover Abstractions

Abstractions that are designed to model either function or data but also incorporate aspects of the other model are termed crossover abstractions. There are very few crossover models, as there are rarely function abstractions that are concerned with data or data abstractions that are concerned with goals. It can be argued that dataflow incorporates goals; however, this is mostly the result of loose definitions and is not part of pure dataflow [Kahn 1974; Johnston 2004]. There are two examples of established crossover abstraction techniques: the Goal-Directed Task Analysis (GDTA) with information requirements [Endsley et al. 2003] and the sensor-annotated abstraction hierarchy [Reising and Sanderson 2002a, b, and c].

4.1. Goal-Directed Task Analysis

Goal-Directed Task Analysis (GDTA) is a functional abstraction technique where nodes represent goals, decisions, and actions [Endsley et al. 2003]. The links between the nodes represent part-whole relationships and in this sense, the GDTA is structurally very similar to the HTA. The GDTA technique inherits most of the HTA’s flexibility and increases the expressiveness with its representation of information requirements and decision questions. The GDTA technique’s ability to represent both functions and data in one coherent abstraction makes it very tractable and applicable to modeling the CBNRE domain, which must focus on both parts. The GDTA technique provides no inherent mechanisms for scheduling or representing parallelism, as is the case with HTA, both of which are of vital interest in the large scale multiple-actor CBNRE domain.

The GDTA is considered a crossover abstraction technique because it augments its nodes with information requirements that ascertain the data required to perform the node’s task and to maintain situational awareness (SA) [Endsley 2001; Endsley et al. 2003]. Situational awareness is the node’s performer’s, actor’s, or in the CBNRE domain the human responder’s understanding of his or her environment and the goal/function relationship to that environment [Endsley et al. 2003]. The concept of situational awareness has swayed between being focused almost exclusively on awareness to a more balanced meaning between situation and awareness [Flach et al. 2004].

The Incident and Hazard Mitigation task is modeled by the GDTA, as presented in Figure 9. Notice that the volume of data represented in the GDTA technique is greater than in any other model depicted thus far. This increase in data is due to expressiveness and crossover nature of the GDTA technique. The GDTA technique presented in Figure 9 is not the standard GDTA technique in that the information requirements have been augmented to include tools and resources, thought processes, and people or groups in addition to the standard information requirements. The Construct Perimeter is not explicitly represented in the GDTA technique, as the GDTA captures the concept of perimeter construction in its decision question: “How can a secure and effective boundary be created between the public and warm and hot zones?” (See the question in the upper left of Figure 9.) This possible reduction in granularity is offset
by the fact that the information requirements represented for the Establish Perimeter task capture all the information presented in the dataflow technique (Section 3), the thought processes presented in the CbTA technique (Section 2.3, Figure 5), and the objects represented in the abstraction hierarchy technique (Section 2.1, Figure 2). Even though the GDTA technique has the largest volume of information, including clearly denoting the decision questions, it does not represent information or process ordering and scheduling as the do dataflow techniques described in Section 3.
4.0 Incident and Hazard Mitigation

How can the risk of the substance be minimized?

How can the hot zone be sealed off in an appropriate fashion?

How can a secure and effective boundary be created between the public and warm and hot zones?

4.1 Establish Perimeter

4.2 Isolation of Contaminants or Hazards

How can the threat be contained and minimized?

Given the hazard what isolation method is appropriate?

Is the isolation method being used in a manner consistent with its guidelines and current constraints?

How long and how well is the contaminant working?

4.3 Mitigate

How can the risk of the substance be mitigated?

Tools and Resources
- Resources to create barrier
- Resources to enforce perimeter
- Resources to monitor perimeter
- Weather service (weather conditions)
- Access Control Points (ACP)

Thought Processes
- Entry/exit procedures
- Qualifications to enter hot zone (from personnel/civilians)
- Use of staging areas
- Use of decontamination lines
- Cordon off area
- Radiation incident address and implement potassium iodide (KI)
- Consider resources being brought to scene
- Be aware of panic/fear
- Site security
- Knowledge of local geography
- Knowledge of local street patterns
- Ensure access control for responders
- Search the area for secondary devices
- Personal Protective Equipment for your first responders

People or Groups
- Law Enforcement
- Fire
- Office of Emergency Management

Information Requirements
- Available and necessary Personal Protective Equipment
- Type of event
- Number of people
- Size of scene
- Location of event
- Type of situation
- Weather and environmental conditions
- Wind direction and speed
- Awareness of secondary devices
- Selected evacuation routes
- Defined predicted hazard area
- Availability of time
- Availability of communication systems
- Availability of personnel
- Availability of vehicles, barricades, and other traffic control equipment
- Pertinent maps, diagrams, and plans
- Simulations at variance with assumptions in plans and procedures

Tools and Resources
- Environmental monitors
- Isolation equipment
- Monitors/detection kits

Thought Processes
- Identification methods for agent
- Types of hazards

People or Groups
- Law Enforcement
- Fire
- HAZMAT
- Civil Support Team
- Bomb Technicians

Information Requirements
- Awareness of evidence preservation needs
- Used equipment
- Identification methods for agent
- Types of hazards

Figure 9: The Incident and Hazard Mitigation task modeled via the Goal Directed Task Analysis.
Flach et. al [2004] suggest creating levels of information requirements to better address the balance between situation and awareness. The levels suggested increase the expressiveness of the information requirements component in the GDTA and provide a means of incorporating a decision ladder in order to represent the information requirements, thereby providing mechanisms for scheduling and parallelism. However, it is unclear whether merely expressing the information requirements with a decision ladder will actually mitigate the overall GDTA’s inability to represent scheduling and parallelism.

The information requirement levels are very similar to the abstraction levels in the abstraction hierarchy presented in Section 2.2. The levels proposed are: functional purpose, functional measurement, functional organization, and physical function [Flach et al. 2004]. The physical function is defined as the logical decomposition of the information flow through the network of functions. This increased expressiveness provided by Flach’s levels borders on incorporating a dataflow model into GDTA; in fact it can be argued that a dataflow model is the most appropriate model to express the concepts outlined at the physical function level. In summary the GDTA, with or without Flach’s extensions, is very applicable to the CBNRE domain, is very expressive, represents very limited partial ordering, and is tractable even though it is verbose.

4.2. Sensor-Annotated Abstraction Hierarchy

The Sensor-Annotated Abstraction Hierarchy is a functional abstraction technique based on the aforementioned abstraction hierarchy [Reising and Sanderson 2002a, b, and c]. Unlike the abstraction hierarchy, the Sensor-annotated abstraction hierarchy is not completed in an abstraction-decomposition space. Instead it has a simple dataflow like model depicted over an abstraction hierarchy to represent how sensory information flows through the abstraction levels [Reising and Sanderson 2002a]. The Sensor-annotated abstraction hierarchy technique is a crossover abstraction because it represents both functions and data in one coherent abstraction. The dataflow adaptation provides some partial order to the elements in the abstraction hierarchy. However, the elements are linked together in a means-end relationship mitigating some of the order provided.

This technique is primarily designed to model a physical system and its defined set of sensors [Reising and Sanderson 2002a, b, and c]. It is not a system composed mostly of humans with an undefined and changing set of information gathering actors, as is present in the CBNRE domain. This mismatch regarding the targeted domain causes the sensor-annotated abstraction hierarchy technique to be untraceable at this scale and for this domain, but it may become relevant at the lowest level. The information gathered regarding the Incident and Hazard Mitigation task could not be modeled via the sensor-annotated abstraction hierarchy with any fidelity or faithfulness to the modeling technique. The
sensor annotated abstraction hierarchy, as its name implies, is designed to model individual sensor values and their relationship to a physical system. The level of detail regarding individual sensors for Incident and Hazard Mitigation has not been gathered from the subject matter experts and the supporting literature does not expound upon them, invariable due to security risks. The example provided below is of a hypothetical unmanned helicopter used for visual reconnaissance task, see Figure 10. The sensor-annotated abstraction hierarchy is similar to the dataflow technique (Section 3) as it depicts the flow of sensory information. The main difference between the dataflow technique and the sensor-annotated abstraction hierarchy is that the node hierarchy in the sensor-annotated abstraction hierarchy technique represents a means-ends relationship, whereas in the dataflow technique the node hierarchy represents a producer-consumer of information relationship. The producer-consumer of information relationship is more applicable as it clearly delineates the effects of sensor performance on subsequent tasks that consume sensor information. Furthermore, because the sensor-annotated abstraction hierarchy technique is based on the abstraction hierarchy it inherits its scheduling and ordering deficiencies, identified in Section 2. In summary, sensor-annotated abstraction hierarchy is not applicable at this stage of development, is fairly expressive, has limited ordering representation, and is not tractable given the current CBNRE knowledge resources.

Figure 10: The sensor-annotated abstraction hierarchy for a hypothetical unmanned helicopter used for visual reconnaissance.
5. Recommendations

Three categories of modeling abstraction techniques, namely functional abstractions, data abstractions, and crossover abstractions, were reviewed with regards to their applicability to developers of CBNRE domain HRI systems, expressiveness of abstraction, representation of timing, and tractability.

Function abstractions focus on goals, tasks, and functions making them easy to understand and facilitate communication with subject matter experts and designers unfamiliar with abstraction modeling. However, functional abstractions provide no inherent mechanisms for scheduling or representing parallelism, both of which are of vital interest in the large scale multiple-actor CBNRE domain. The three functional abstractions reviewed (Hierarchical Task Abstraction, WDA’s abstraction decomposition or abstraction hierarchy, and CbTA’s decision ladder and statecharts) exemplify functional abstractions. Between these three abstractions it is debatable whether the HTA technique or the abstraction-decomposition technique is better suited. Due to the aforementioned scheduling and parallelism issues, neither technique is well suited for performing an abstraction in the CBNRE domain. The modeling abstraction goal is to facilitate the designers in developing HRI systems. These systems will operate in parallel with the existing CBNRE response and will require an understanding of task and data scheduling. Furthermore, it is likely that the robots will be used as information providers; therefore, explicit representation of information, its flow, and affect on the CBNRE domain is necessary to understand the impact and the benefit the HRI system will provide. It is for all these reasons functional abstractions are not recommended for informing the design for this domain.

Data abstractions were designed to embrace parallelism by focusing on the data and executing instructions as soon as their local data becomes available. Dataflow, the primary model used in data abstractions, has established extensions that allow for partial or complete scheduling. These features address the outstanding issues with functional abstractions. Furthermore, because of the data abstractions’ focus on the data required at each execution node, identifying weak links and addressing error tolerance due to faulty data is clearer. These weak links in the information flow for CBNRE could be “hot” places for HRI systems to improve that information flow and thereby improve the overall CBNRE response. Dataflow has also been extended to understand data history or groups of data via Multidimensional Synchronous Dataflow (MDSDF) modeling. The disadvantage of dataflow models is that they deemphasize goals and at worse ignore goals altogether. Goals and priority determine how the limited resources in the CBNRE domain are scheduled. However, if the goals are represented as control signals to SWITCH and SELECT elements in a MDSDF then it becomes a clean and effective modeling technique recommended for representing the complexity of the CBNRE domain.

The remaining limitation of MDSDF is that compared to Goal Directed Task Analysis it does not clearly represent what the decision(s) are regarding a task. This can be mitigated by extending MDSDF
slightly by including a Goal Directed Task Analysis style decision question as part of each execution node, see Figure 11. This simple extension permits MDSDF to overcome its remaining representation limitation.

<table>
<thead>
<tr>
<th>Type of event</th>
<th>Available and necessary Person Protective Equipment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location of event</td>
<td>Traffic control equipment</td>
</tr>
<tr>
<td>Size of scene</td>
<td>Type of event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of situation</th>
<th>Number of people</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather and environmental conditions</td>
<td>Pertinent maps, diagrams, and plans</td>
</tr>
<tr>
<td>Simulations at variance with assumptions in plans and procedures</td>
<td>Wind direction and speed</td>
</tr>
<tr>
<td>Type of situation</td>
<td>Awareness of secondary devices</td>
</tr>
<tr>
<td>Availability of time</td>
<td>Selected evacuation routes</td>
</tr>
</tbody>
</table>

Figure 11: The Multidimensional Synchronous Dataflow technique with the addition of a Goal Direct Task Analysis style decision question for Construct Perimeter sub-task.

Crossover abstractions are hybrids between functional and data abstractions. Two techniques were reviewed: the GDTA and Sensor-annotated abstraction hierarchy. The GDTA is a functional abstraction that incorporates data modeling via information requirements. These information requirements can be modeled according to different abstraction levels, which can incorporate full data flow modeling. However, this approach is really two modeling methods loosely coupled, whereas the MDSDF technique represents the CBNRE domain in one model. The GDTA is still a functional abstraction that provides less expressiveness than a data abstraction and therefore has more to overcome as a crossover abstraction.
technique. It is unclear that these crossover features will fully mitigate the scheduling and parallelism issues and may not display the “weak links” as clearly as would the MDSDF technique.

The Sensor-annotated abstraction hierarchy is designed for defined physical domains and is therefore a mismatch for modeling the CBNRE domain. The Sensor-annotated abstraction hierarchy might be useful in modeling a HRI system, but not the CBNRE task in which the system is only a part. For all the reasons stated above neither crossover abstraction is recommended.

There is one recommended modeling abstraction technique recommended for this work related to HRI systems for the CBNRE domain. The recommended modeling abstraction is a data abstraction based on Multidimensional Synchronous Dataflow. Goals will be represented as control signals to explicitly represent goals as the driving force behind scheduling choices. Execution nodes will be extended with GDTA style decision questions to clearly denote the decisions that comprise a task. This abstraction technique will meet the needs for use by developers of CBNRE domain HRI systems by providing an expressive abstraction, rich representation of timing, and design tractability.
References:


