MODELING WING LEVEL OPERATIONS USING FORMAL OBJECT MODELS

Thomas C. Hartrum, Robert J. Hunt, and Michael D. Sarchet

Air Force Institute of Technology
Department of Electrical and Computer Engineering
Wright-Patterson Air Force Base, OH 45433-7765

Abstract

This paper describes the application of *Knowledge Based Software Engineering (KBSE)* techniques to model an Air Force fighter wing's mission performance under differing personnel assignments and levels of automation [4] [10]. A formal domain model of the mission of an Air Force fighter wing was developed and the feasibility that such a model can be used to determine the effect of automation on unit readiness and unit was demonstrated. A prototype tool was written in Ada to demonstrate assessing the impact of both worker and automated tool assignments on the overall performance of processing a complex task. Additionally, since the resulting task model not only models the general behavior of an Air Force fighter wing, but with an appropriate detailing of the specific tasks, workers, and tools, along with the rules for determining task performance, the model developed here could find application in many domains.

1 Introduction

Many areas of the Air Force benefit from the use of automation and data processing. Such tools are beginning to find their way into operational units, such as a fighter wing. In planning for such systems, a method is needed to assess the impact of automating various portions of a unit's operations on its overall mission effectiveness. This research investigates the application of *Knowledge Based Software Engineering (KBSE)* techniques to model a unit's mission performance under differing personnel and automation assignments [4] [10].

The specific goal of this research is to develop a formal domain model of the mission of an Air Force fighter wing and to demonstrate that such a model can be used to determine the effect that automation has on unit readiness and unit effectiveness. This involves formally modeling both the unit mission (behavior) and existing and proposed software tools, and defining the mappings needed to allow reasoning about the effects of the latter on the former.

The problem addressed in this research is somewhat different from those typically attacked using formal modeling and domain analysis. Usually the software to be

developed is modeled, with the goal of transforming that model into the final executable software. In this case, the interest was more in modeling the *performance* of software as well as the behavior and performance of the real (not software) fighter wing in the execution of its mission. Thus this effort involves using formal object oriented domain modeling techniques to develop a *process* model rather than the more traditional use to develop a *product* model.

2 Domain Analysis

The overall system to be modeled is one in which specific resources, particularly personnel and automated tools, are applied to a complex workload of Command, Control, and Communication (C^3) tasks in an Air Force fighter wing. The domain analysis is approached in two phases: a high level model of assigning resources to tasks, and a more detailed model of the resources and tasks themselves.

2.1 Tasking Model

At a high level of abstraction, the wing mission can be modeled as the assignment of Air Force personnel and automated tools to perform tasks that must be accomplished, and the assessment of the resulting unit effectiveness and efficiency. An informal structural model of this system based on Rumbaugh [9] is shown in Figure 1. This is augmented by a formal model using the formal language "Z" (zed) [3]. The pertinent parts of this model are discussed in more detail in the following paragraphs.

2.1.1 Task

An instance of a *task* represents a piece of work that must be done by a single person, *i. e.* an indivisible task. The task's *afsc* and *skill level* specify the type of desired qualifications needed to perform this task properly, and when assigned to a person with those qualifications the *nominal time* indicates the expected amount of time required to complete the task. The *inputs* and *outputs* specify the data needed to begin the task and the data produced by the task respectively. The *precedes* association specifies any required ordering of tasks, that is any tasks that must

Figure 1: Assignment of tasks.

complete before the next one can start. If a pair of tasks (t_1, t_2) is in the *precedes* association, then task t_1 must be completed before task t_2 can be started. Furthermore, if task t_1 precedes task t_2 (that is, $(t_1, t_2) \in \text{precedes}$) then at least one of t_1 's outputs must be an input to t_2 . The corresponding *Z* schemas are as follows.

<i>TaskSet</i> $\text{task_set} : P \text{ Task}$

<i>Task</i> $\text{name} : \text{seq CHAR}$ $\text{afsc} : \text{AFSC_TYPE}$ $\text{skill_level} : \mathcal{N}$ $\text{status} : \text{TASK_STATUS}$ $\text{inputs} : P \text{ DATA}$ $\text{outputs} : P \text{ DATA}$ $\text{nominal_time} : \text{TIME}$
--

<i>Precedes</i> $\text{precedes} : \text{Task} \leftrightarrow \text{Task}$ $\forall (t_1, t_2) : \text{precedes} \bullet$ $t_1.\text{outputs} \cap t_2.\text{inputs} \neq \{ \}$
--

2.1.2 Worker

A *worker* is primarily characterized by the static attributes *afsc* and *skill level* which specify the qualifications of this person, and the dynamic attribute *availability* which is used in the executable model to determine when a worker is busy with a task.

<i>Worker</i> $\text{name} : \text{seq CHAR}$ $\text{afsc} : \text{AFSC_TYPE}$ $\text{skill_level} : \mathcal{N}$ $\text{availability} : \text{BOOLEAN}$
--

2.1.3 Assignment

The association *assignment* represents the assignment of a specific task to a specific person. Associated with each such assignment is the actual *time* that the task will require with this person and the resulting *accuracy*, both attributes of the association. In the current prototype implementation these are automatically determined from the nominal time, based on a comparison of the skill level of the task and the assigned person. The worker skill level must be equal to or greater than the required skill level of any task assigned to that worker and the *afsc* of a worker must match the *afsc* of any task assigned to that worker.

<i>AssignmentAttr</i> $\text{accuracy} : \mathcal{N}$ $\text{time} : \text{TIME}$ $\text{accuracy} \leq 100$

<i>Assignment</i> $\text{assignment} :$ $(\text{Task} \times \text{Worker}) \rightarrow \text{AssignmentAttr}$ $\forall (t, w) : \text{dom assignment} \bullet$ $t.\text{skill_level} \leq w.\text{skill_level}$ $\forall (t, w) : \text{dom assignment} \bullet t.\text{afsc} = w.\text{afsc}$
--

2.1.4 Tool

A *tool* represents a form of automation, such as a software program. The *supports* association indicates which tools support which tasks, and the extent of that support (via the association attribute *level*). Association *assigned*, on the other hand, indicates that a specific tool has been assigned to be used on a task by the appropriate worker. In the prototype implementation, the time and accuracy of task performance is improved if a tool is assigned and available, based on the level of support defined.

<i>Tool</i> $\text{name} : \text{seq CHAR}$ $\text{availability} : \text{BOOLEAN}$
--

<i>Assigned</i> $\text{assigned} : \text{Task} \leftrightarrow \text{Tool}$
--

<i>SupportsAttr</i> $\text{level} : \mathcal{N}$

<i>Supports</i> $\text{supports} : (\text{Tool} \times \text{Task}) \rightarrow \text{SupportsAttr}$

Figure 2: System Dynamic Model.

2.1.5 System Behavior: The Dynamic Model

Rumbaugh characterizes an object's behavior via the *dynamic model*, described in this section, and the *functional model*, described in Section 2.1.6 [9]. The dynamic model is a state-model. An informal dynamic model is shown in Figure 2. A *formal* dynamic model can be defined using *Z* schemas to specify *states* and *events*, along with a *state transition table*. This is illustrated as follows for the *Worker* and *Tool* object classes.

Worker Dynamic Model The worker is either idle or busy. When commanded to start a task, the worker selects a tool and issues a *StartTool* command. When the task is completed, the worker issues a *StopTool* command and a *TaskDone* message.

State Name: Idle

State Description: Worker is idle and available for assignment.

Idle _____
Worker _____
availability = *TRUE*

State Name: Busy

State Description: Worker is working on a task.

Busy _____
Worker _____
availability = *FALSE*

Event Name: StartTask(task)

Event Description: Causes worker to begin working on specified task.

StartTask _____
task : *Task*

Event Name: TaskFinished

Event Description: Task has reached finished state.

TaskFinished _____

State Transition Table:

Current	Event	Next	Action
Idle	StartTask	Busy	StartTool
Busy	TaskFinished	Idle	StopTool; TaskDone

Tool Dynamic Model The tool is either available or in use. If available, it will be assigned to the first worker to issue a *StartTool* event, and will belong to that worker until a subsequent *StopTool* is issued, when it will again be considered available.

State Name: Available
State Description: Tool is not in use and can be assigned.

<i>Available</i>
<i>Tool</i>
<i>availability = TRUE</i>

State Name: InUse
State Description: Tool is being used.

<i>InUse</i>
<i>Tool</i>
<i>availability = FALSE</i>

Event Name: StartTool
Event Description: Tool is assigned for use.

<i>StartTool</i>

Event Name: StopTool
Event Description: Tool is released.

<i>StopTool</i>

State Transition Table:

Current	Event	Next	Action
Available	StartTool	InUse	
InUse	StopTool	Available	

2.1.6 System Behavior: The Functional Model

Rumbaugh represents the functional model using traditional data flow diagrams (DFDs). For example, when the *Scheduler* (shown in Figure 2) receives a *TaskDone(task)* event, it performs the *update_task_status* action as follows. For the specified task that just finished, all tasks that depend on it (each task that the finished one *precedes*) are checked. If a task is in the *Waiting* state (*status = waiting*) and all of its predecessors are *finished* then it is set to *ready*, otherwise it is left in the *Waiting* state. This is shown in Figure 3 for the top level DFD and one level decomposition. The functional model is formalized using *Z* dynamic schemas for the leaf processes, with pre- and post-conditions specified in predicate logic. This is illustrated as follows for the leaf processes in Figure 3.

<i>GetDependentTasks</i>
<i>Precedes</i>
<i>task? : Task</i>
<i>dependent_tasks! : PTask</i>
<i>dependent_tasks! = ran({task?}) < precedes</i>

Figure 3: Functional Model.

<i>GetReadyTasks</i>
<i>Precedes</i>
<i>dependent_tasks? : PTask</i>
<i>ready_tasks! : PTask</i>
<i>ready_tasks! = {t : Task </i> <i>t ∈ dependent_tasks? ∧ (∀ s : Task</i> <i>• (s, t) ∈ precedes ⇒ s.status = finished)}</i>

<i>SetTasksToReady</i>
<i>ΔTaskSet</i>
<i>ready_tasks? : PTask</i>
<i>#task_set' = #task_set</i>
<i>∀ t' ∈ task_set' ∃ t ∈ task_set </i> <i>(t ∉ ready_tasks? ⇒ (t' = t))</i> <i>t ∈ ready_tasks? ⇒</i> <i>(t'.name = t.name</i> <i>t'.afsc = t.afsc</i> <i>t'.skill_level = t.skill_level</i> <i>t'.status = ready</i> <i>t'.inputs = t.inputs</i> <i>t'.outputs = t.outputs</i> <i>t'.nominal_time = t.nominal_time))</i>

2.2 Detailed Domain Model

The task model presented in Section 2.1 models the general behavior of an Air Force fighter wing. However, a closer examination of this model shows that it supports a much wider domain. Almost any Air Force unit (or *any* organization) could be viewed as assigning tasks to workers. The next level of domain modeling involved a more specific definition of the appropriate tasks, workers, and tools, along with the rules for determining task performance accuracy and time for a fighter wing. This phase depended heavily on prior models of a wing [6][7][8] and discussions with wing domain experts.

2.2.1 Scoping the Domain

It was obvious early on that modeling an entire fighter wing was an extremely large effort. Therefore, in order to demonstrate feasibility, the scope was reduced based largely on suggestions by the sponsor. The *Worker* portion of the model was limited to the Wing Operation Center's Mission Planning Cell, while the *Task* portion of the model was limited to processing an Air Tasking Order (ATO) from receipt of the ATO FRAG to the completed mission. As this task was further analyzed, the scope was further reduced, first to preparing mission plans and schedules, and then to determining mission specifics. No attempt was made to explicitly model specific automation tools available at a fighter wing. Rather a general purpose "tool" was defined for use in the feasibility model. Finally, defining the rules for assessing the impact of particular workers and tools on the completion of a task, and defining the mission effectiveness and efficiency metrics turned out to be extremely difficult. Simple models were used to demonstrate feasibility of the approach. Each of these areas is discussed in more detail in the following sections.

2.2.2 Workers

Analysis of the wing's organization from a wartime perspective resulted in the model shown in Figure 4. All per-

Figure 4: Wing Organization from Wartime Perspective.

sonnel are modeled as having an AFSC and Skill Level.

2.2.3 Tasks

The ATO task was modeled as a set of subtasks that are specializations (in the Rumbaugh sense [9]) of a general *ATO Task*, as shown in Figure 5. Thus each assignable task inherits the attributes as shown. The set of subtasks, as well as their inputs and outputs, were determined by performing a functional decomposition of the overall task *Conduct Wing C² Operations*. Figure 6 illustrates the result for the decomposition of function *Determine Mission*

Figure 5: Air Tasking Order Task Model.

Specifics into the final set of tasks included in the feasibility model. Higher level decompositions leading to this figure are documented by Hunt and Sarchet [4][10].

An important aspect of this model is the set of values for the task attributes of *afsc*, *skill level*, and *nominal time* for the specific tasks involved in processing an ATO. These values were determined based on conversations with domain experts.

2.2.4 Assessments

As shown in Figure 1, an important domain-specific part of the model is the *assignment* association that represents the assignment of a task to a person. In particular, rules are needed for determining the time to complete the task and the accuracy of the results for all feasible pairings of personnel and tasks. This information was not readily available, so in the feasibility model a simple weighting factor is used, based on whether the task and worker *afsc* match, and the difference between their skill levels. If a tool is assigned to the task, the values are adjusted accordingly. The level of tool support for a particular task was arbitrarily defined for demonstration purposes. The details of these approximations are explained by Hunt [4].

In order to assess the impact of various personnel and tool assignments on the overall mission (of processing an ATO), several other domain factors need to be known. The *precedes* association is necessary to determine the impact of assigning too many tasks to the same worker and not taking advantage of potential concurrency in the system. This information was extracted automatically from the functional model shown in Figure 6. A precedence graph was formed, as shown in Figure 7, and the *precedes* mapping function derived from that.

The overall goal of the model is to assess the effect of worker and automated tool assignments on overall unit effectiveness and efficiency. However, clear definitions of the measurement of unit effectiveness and efficiency were not available. To demonstrate the feasibility of this approach, the following metrics were defined based on a general assessment of a tasking system [10].

- **Time to Complete Mission:** This defines the wall

Figure 6: Functional Model of *Determine Mission Specifics*.

Figure 7: Dependency Graph for *Determine Mission Specifics*.

clock time from when the first task starts until the last task is finished. It basically reflects how long the given assignment of workers and tools takes to complete the ATO processing.

- **Total Work Time:** This is the sum of the times to perform each task. It reflects how long the ATO processing would take if all tasks were performed sequentially.
- **Mission Accuracy:** This is an estimate, on a scale of one to ten, of how accurate (error free) the overall products of the ATO processing are. The feasibility model calculates this as the average of the accuracies of the individual tasks as performed by the assigned workers.
- **Worker Efficiencies:** This is a measurement of the percent of time that each worker was busy processing the ATO. Since only the ATO task is modeled, workers are considered idle when not working on an assigned task. This indicates how well workers are being utilized and what workers are available to offload the busier ones.
- **Tasks Ready When Worker Wasn't:** This indicates how many times a task was ready to be executed (*i. e.* all of its input data was available) but the assigned worker was busy with another task. This indicates an assignment of personnel where one or more persons are the bottleneck.
- **Critical Path:** This determines which tasks were on the critical path, and indicates which workers were responsible for the critical path.

2.3 Prototype Tool

In order to demonstrate the feasibility of the approach taken and of using an executable model, a prototype tool was written in Ada. The tool consists of two parts.

The task assignment portion of the tool presents each of the ATO tasks along with a list of available personnel and allows the user to assign workers and tools to each task [4]. The tasks's required AFSC and skill level are listed, along with those of the available workers. A list of tools, along with their level of support for this task, is also presented and the user is allowed to select one tool to aid the worker. Assignments are interactively specified by the user for each task.

The performance analysis portion of the tool analyzes the mission performance in terms of the metrics defined in Section 2.2.4 [10]. The prototype tool performs this analysis by executing a discrete-event simulation of the task processing derived from the dynamic model shown in Figure 2, and accumulating statistics about task and worker utilization and times.

3 Summary of Status

The work by Hunt [4] and Sarchet [10] has provided a good start at understanding the goals of this research effort and

an approach to meeting those goals. The prototype tool demonstrates the possibilities for assessing the impact of both worker and automated tool assignments on the overall performance of processing a complex task. In addition, this work demonstrated the difficulty of determining detailed domain information by non-domain experts, especially when the domain experts aren't locally available for daily contact.

Additionally, the task model in Section 2.1 not only models the general behavior of an Air Force fighter wing, but with an appropriate modeling of the specific tasks, workers, and tools, along with the rules for determining task performance, could model any Air Force unit or other organization. Thus the model developed here could find application to many domains, including, for example, modeling the software development process in support of the Software Engineering Institute's Process Maturity Model [1].

4 Future Plans

While the initial prototype has demonstrated the feasibility of applying KBSE techniques to modeling an operational unit's effectiveness, more work is needed to show the advantages of formal modeling in a real environment. The following are some of the planned extensions to this work.

- A more detailed domain analysis is needed to determine the proper AFSC and skill levels for mission tasks and fighter wing personnel.
- The scoped domain model and prototype tool demonstrate very useful results for a single complex task (ATO processing) that is statically assigned. More interesting would be to consider several independent tasks, with the possibility of a new complex task being assigned after work has begun on one or more other tasks.
- The approach of assessing worker assignments to tasks by comparing AFSC and skill level appears to be a good idea, but a more accurate way of determining the resulting time and accuracy metrics is needed, specifically for fighter wing type tasks. A formal rule base should provide tailorable definitions of this relationship.
- The current approach of defining tool support for a specific task is too simplistic. A more realistic model needs to be developed, as well as a means for determining the impact of using a tool on a task's time and accuracy. This is an area of particular interest to the sponsor.
- Better metrics for assessing the overall mission performance are needed. The sponsor suggested *unit effectiveness* and *unit readiness* but did not provide a detailed definition or means of calculating them [5]. The metrics used in the prototype have merit, but need to be better defined and tailored to fighter wing missions.

- The high level model of assigning tasks to workers while trying to optimize some criteria (*e. g.* task completion time) subject to various constraints is a well known problem in the operations research area [2]. Although this problem in general is known to be np-complete, there are many heuristics in the operations research field for making “good” assignments that should be explored. This would allow automatic assignment of workers and software tools to any tasks needing to be done.

of Technology(AETC). Wright-Patterson AFB. OH. December 1994.

References

- [1] *Capability Maturity Model for Software*. Technical Report CMU/SEI-91-TR-24. Software Engineering Institute. August. 1991.
- [2] French. Simon. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. New York: John Wiley & Sons. 1982.
- [3] Hartrum. Thomas C. and Paul Bailor. *A Formal Extension to Object Oriented Analysis Using Z*. Technical Report AFIT/EN/TR94-07. Air Force Institute of Technology. October 7. 1994.
- [4] Hunt. Robert. *Modeling Operational Task Assignment in Air Force Wing Command and Control*. MS thesis. AFIT/GCS/ENG/94D-09. Graduate School of Engineering. Air Force Institute of Technology(AETC). Wright-Patterson AFB. OH. December 1994.
- [5] Langloss. Randel K. *Knowledge Based Software Engineering (KBSE) Support: A Formal Model of Wing-Level C2 Applied to the 432nd Fighter Wing Misawa AB. Japan*. Technical Report. HQ PACAF/SCC. June 14. 1993.
- [6] MITRE. *Theater Battle Management Command. Control. Communications. Computer. and Intelligence (TBM C4I) Architecture Air to Air (F-15) Wing Operations Center (WOC)*. Technical Report. HQ ACC/DRI. 1 November 1993.
- [7] MITRE. *Theater Battle Management Command. Control. Communications. Computer. and Intelligence (TBM C4I) Architecture Air to Ground (F-16) Wing Operations Center (WOC)*. Technical Report. HQ ACC/DRI. 1 November 1993.
- [8] RCA. Government Systems Division. *TAF Mission. Level. Functions Analysis: Volume I - Mission Related Functional Analysis*. Technical Report. TAC. February 1978.
- [9] Rumbaugh, James, et al. *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall. 1991.
- [10] Sarchet, Michael. *Modeling Workload Effectiveness and Efficiency of Air Force Wing Command and Control*. MS thesis. AFIT/GCS/ENG/94D-22. Graduate School of Engineering, Air Force Institute