The Common Prototyping Environment

A Framework for Software Technology Integration, Evaluation, and Transition

Mark H. Burstein and Richard Schantz, BBN Systems and Technologies
Marie A. Bienkowski and Marie E. desJardins, SRI International
Stephen F. Smith, Carnegie Mellon University Robotics Institute

The ARPA/Rome Planning Initiative is organized in part around the notion of a Common Prototyping Environment. The CPE is designed to

- give ARPA's researchers and developers access to a common body of support software, development tools, and sharable domain knowledge and data, thereby fostering the development of reusable planning and scheduling technologies;
- support experiments that integrate these technologies to solve realistic planning problems; and
- ease the transition of these technologies into operational prototypes through demonstrations of interoperability.

The first major element of the CPE to be developed was the CPE Repository, an electronic clearinghouse for software (sources and executables) developed under ARPA auspices, and for data and knowledge about military transportation planning.

The second element was the CPE Testbed, a suite of software tools that supports the integration of, experimentation with, and evaluation of prototype planning and scheduling systems. The CPE Testbed grew out of ARPA's first set of Technology Integration Experiments (TIEs), and brings together the systems and tools used in the TIEs so that they work in the same distributed environment, communicating among themselves using a consistent set of distributed software communications protocols and using a single knowledge-representation language for information exchange.

BBN Systems and Technologies and the ISX Corporation have been developing and supporting the CPE. BBN has had primary responsibility for developing the CPE and some of the supporting software used in integration experiments, and ISX has had primary responsibility for providing access to domain data and knowledge. In this article we'll explain the development of the CPE Repository, the TIEs, and the CPE Testbed, and how they advance the ARPA vision of distributed, collaborative planning.

The CPE promotes the development of collaborative, distributed planners by combining a repository for shared software and data, integrated software systems, and a testbed for experimentation.

The CPE repository

From the outset, ARPA promoted the sharing of data and software development tools among its R&D contractors, both to better focus on a common set of domain-specific problems, and to reduce redundant tool building. Toward this end, we established the CPE Repository as the primary means of collecting and disseminating data, source code for reusable development software tools, demonstration prototype software systems, and software collected or developed by ARPA's contractors. The repository is housed on a Unix workstation and accessed over the Internet by FTP.

First, to avoid problems that arise when software systems developed in different places are integrated, the contractors...
established community standards for the basic hardware and software underpinnings of the prototype software to be developed. These standards covered the type of Unix workstations and the version of Common Lisp to be used, and included adoption of the Common Lisp Interface Manager for user-interface development. The contractors also set standards for such things as X Windows and TCP/IP. On top of these basic necessities for software development, the contractors gradually added a range of software tools, including:

- graphical interface tools (knowledge-base network browsers, a scientific-graphics plotting package, and a map-display system),
- system-development tools (a standard Lisp Defsystem and logical path names),
- a knowledge-base maintenance system (Loom),
- a knowledge-directed database-access mechanism compatible with that system (LIM/IDI),
- a knowledge-based systems-intercommunication protocol (KQML), and
- an object-oriented distributed-communications mechanism that works with many hardware platforms and programming languages (Cronus).

Using these and other tools from their own laboratories, the contractors developed a number of other technology packages and installed them, with documentation, in the CPE Repository. Some contributions were primarily technology demonstrations, while others were intended for use in other development efforts. Among the contributions were generative planners, temporal-information maintenance systems, constraint-based-scheduling systems, intelligent database-access and database-query planning systems, decision support and analysis tools, a knowledge-based-systems development tool for reasoning with uncertainty, and a Lisp-based statistics and metering package to support quantitative experimental analysis of AI systems. Table 1 lists the tools and packages that make up the CPE Repository.

As a clearinghouse for domain knowledge and data about military transportation planning, the repository includes a large body of textual materials about current military planning procedures and sample problems derived from officer’s training courses on the subject. It also includes data files containing a variety of sample data sets, taken primarily from unclassified military-training materials and scenarios used in ARPI’s Integrated Feasibility Demonstrations (IFDs, detailed in the article on page 27).

**K R S L and the shared-domain ontology.**

ARPI also sought to promote a consistent way of accessing all the data and the different kinds of representations of objects, actions, time relations, and plans that are inputs to or products of military planning. As a result, during the project’s second year, many ARPI members jointly developed the Knowledge Representation Specification Language (KRL, pronounced “carousel“). KRL uniformly describes and stores the domain data in the repository. In addition to basic object and concept descriptions, KRL includes forms for units of measure, general relations and propositions, temporal relations, plan and goal descriptions, and producer-consumer constraints.

On top of this basic definitional syntax, ARPI members developed substantial fragments of a uniform shared-domain ontology, which describes the data elements and object types that are inputs to or outputs of the systems that are coupled to solve sample planning problems. This ontology is a vocabulary of objects and relationships spanning the

Table 1. Some of the software tools and "technology packages" in the CPE Repository.

<table>
<thead>
<tr>
<th>Name</th>
<th>Developer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cronus</td>
<td>BBN</td>
<td>Object-oriented distributed-communications mechanism</td>
</tr>
<tr>
<td>KQML</td>
<td>Paramax (now Unisys)</td>
<td>Knowledge-based systems-intercommunication protocol</td>
</tr>
<tr>
<td>Loom</td>
<td>USC/ISI</td>
<td>Knowledge base maintenance system</td>
</tr>
<tr>
<td>LIM</td>
<td>Paramax (now Unisys)</td>
<td>An intermediate layer that connects Loom to IDI</td>
</tr>
<tr>
<td>IDI</td>
<td>Paramax (now Unisys)</td>
<td>A knowledge-directed database-access mechanism</td>
</tr>
</tbody>
</table>

Technology packages:

<table>
<thead>
<tr>
<th>Name</th>
<th>Developer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clasp</td>
<td>Univ. of Massachusetts</td>
<td>A Lisp-based statistics and metering package</td>
</tr>
<tr>
<td>CoBase</td>
<td>Univ. of California at Los Angeles</td>
<td>An intelligent database-access and database-query planner</td>
</tr>
<tr>
<td>Demos</td>
<td>Rockwell</td>
<td>Decision support and analysis tool</td>
</tr>
<tr>
<td>DT</td>
<td>Rockwell</td>
<td>Decision support and analysis tool</td>
</tr>
<tr>
<td>O-Plan2</td>
<td>Univ. of Edinburgh</td>
<td>A generative planner</td>
</tr>
<tr>
<td>Primo</td>
<td>General Electric R&amp;D Center</td>
<td>A knowledge-based systems-development tool for reasoning with uncertainty</td>
</tr>
<tr>
<td>SIMS</td>
<td>USC/ISI</td>
<td>An intelligent database-access and database-query planner</td>
</tr>
<tr>
<td>SIPE-2</td>
<td>SRI</td>
<td>A generative planner</td>
</tr>
<tr>
<td>Tachyon</td>
<td>General Electric R&amp;D Center</td>
<td>A temporal reasoner</td>
</tr>
<tr>
<td>TMM</td>
<td>Honeywell</td>
<td>A temporal reasoner</td>
</tr>
</tbody>
</table>
different transportation resources, ports, equipment, and other data structures that participate in planning. It originally handled the information involved in the kinds of planning problems encountered during ARPI's second Integrated Feasibility Demonstration (IFD-2). Recent attempts to extend the ontology have focused on noncombatant-evacuation operations, as in IFD-3, and on air-campaign planning.

As mentioned earlier, the repository includes a large volume of data in its original textual and fixed-format ASCII forms. To make this data more uniformly available, selected subsets were translated into KRSL using the concepts in the ontology, so that researchers and developers could draw on a consistently described reservoir of domain data, and could use a consistent vocabulary for that data. This standardization began during the TIEs, and was fully realized in the CPE Testbed. In the testbed, all communications between modules (planning and scheduling systems) are TCP/IP-based messages in terms of KRSL and the ontology’s vocabulary.

### Table 2. Technology Integration Experiments.

<table>
<thead>
<tr>
<th>Type of TIE</th>
<th>Developers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Infrastructure</td>
<td>BBN, ISX, SRI, and Paramax (now Unisys)</td>
<td>TIE 1 demonstrated the knowledge server and the distributed, knowledge-based communications elements of the Common Prototyping Environment.</td>
</tr>
<tr>
<td>2. Case-based reasoning, generative planning</td>
<td>General Electric R&amp;D Center, and SRI</td>
<td>TIE 2 used a case-based reasoner to support a generative planner in a specialized reasoning task selecting force units for a military operations plan.</td>
</tr>
<tr>
<td>3. Constraint-based scheduling, deployment planning</td>
<td>Carnegie Mellon Univ., BBN, and SRI</td>
<td>TIE 3 demonstrated two different roles for a constraint-based scheduling system in a heterogeneous military planning system. TIE 3a applied deployment constraints to the final stages of deployment-plan development, thus producing a feasible deployment plan. TIE 3b used the scheduler to perform preliminary deployment-plan analysis during the initial phases of course-of-action development, to filter the options for an operations plan.</td>
</tr>
<tr>
<td>4. Temporal reasoning, generative planning</td>
<td>General Electric R&amp;D Center, Honeywell, and SRI</td>
<td>TIE 4 developed a generic interface between two temporal reasoners and a generative planner such that either reasoner could propagate the temporal constraints in a developing plan. This arrangement enabled comparative experimentation with the reasoners.</td>
</tr>
<tr>
<td>5. Temporal reasoning, case-based force expansion</td>
<td>General Electric R&amp;D Center, and BBN</td>
<td>TIE 5 applied temporal constraint management during case-based force selection and expansion, where selected forces are &quot;unpacked&quot; to form elements of a detailed deployment plan.</td>
</tr>
<tr>
<td>6. Case-based reasoning, force expansion</td>
<td>General Electric R&amp;D Center, BBN, and Mitre</td>
<td>TIE 6 incorporated the functionality of the prototype force expander from IFD-2 into a more general facility for case-based force expansion. TIE 6 used techniques for refining and enumerating the elements of forces selected during high-level planning, together with case-based reasoning techniques for tailoring both canonical and case-specific force descriptions to new situations.</td>
</tr>
</tbody>
</table>

### Technology Integration Experiments

In the spring of 1992, following IFD-2, planning began for a series of Technology Integration Experiments that would develop and demonstrate interoperability among emerging ARPI technologies, collect information about trade-offs between technical alternatives, and validate various technologies for use in IFDs. The initial set of TIEs focused on integrating more ARPI research technologies into a distributed software environment that included the IFD-2 prototypes. These TIEs addressed related transportation scheduling and planning problems grounded in the IFD-2 scenario. As a group, the TIEs demonstrated most of the elements of infrastructure and planning technology to be introduced into the CPE over the succeeding two years.

The TIEs were explicitly experimental; they explored methods of combining technical components that played specific roles in planning or problem solving, and tested designers' claims for their technology's applicability. When possible, we sought to contrast mechanisms providing the same or similar functions. For example, in one TIE, two systems for temporal-constraint maintenance interoperated with a plan-generation system during course-of-action generation. By sharing a standard interface, the two systems could be directly compared performing the same task. In other cases, TIEs investigated pairs of technologies that might mutually benefit from interoperability.

Each TIE included a "final exam" that measured the performance of the technical components against baseline criteria. We conducted these exams to identify and prepare component technologies for insertion into an IFD, and to benchmark performance on typical military planning and scheduling problems.

Table 2 lists the TIEs that have been completed to date; the first four were completed in 1992 and contributed to the development of the initial CPE Testbed. Figure 1 places all six TIEs within a functional model of military plan development. At the top is TIE 1, the Infrastructure TIE, which laid the
groundwork needed for components of the planning process to uniformly access and communicate information about the plans under construction.

Figure 1 also represents the three main functional areas of plan development: course-of-action generation, deployment plan expansion, and plan analysis. Course-of-action generation, the first major focus of the TIEs, involves strategic and high-level tactical planning for employment and deployment of forces. In the TIEs, most of the activity relating to course-of-action generation was modeled using the System for Operations Crisis Action Planning (Socap), a plan generator that was the focus of IFD-2 and that was built on top of the SIPE-2 generative planner. Different TIEs examined how to improve the plans produced by Socap by exploiting three kinds of external functionality: temporal-constraint management (TIE 4), case-based reasoning for force selection (TIE 2), and constraint-based reasoning for transportation-resource analysis (TIE 3b). We used the temporal reasoners and constraint-based schedulers as plan-analysis tools during course-of-action generation. These constraint managers also contributed to the final plan by propagating constraints and dependencies and representing their effects more explicitly.

The other major focus in the TIEs was on deployment plan expansion (also called force expansion), the process of producing detailed deployment plans from high-level inputs about major force groups to be deployed (the output of course-of-action generation). TIE 3a used constraint-based scheduling during deployment planning to develop reasonable deadlines and choices for the mode of transport (air, sea) for the forces. TIEs 5 and 6 involved the use of case-based reasoning to structure tailored forces for specific missions, and to reason about the temporal constraints in deployment of those forces.

The infrastructure TIE. The goal of the Infrastructure TIE was to demonstrate integration and interoperability technology for the Common Prototyping Environment in particular, and for new ARPI software in general. The Infrastructure TIE was developed under the premise that knowledge-based software systems could interoperate most ef-
ffectively with other software systems in a
distributed environment by using a consist-
ent representation language as an interlin-
gua. They could thereby avoid having to de-
develop and maintain independent means of
translating messages between every pair of
communicating systems. The TIE demon-
strated that this integrated approach to inter-
agent communication would work for the
sharing of knowledge among knowledge-
Based software agents, and for knowledg-
level access to common databases. It formed
the basis for the first working prototype of
the CPE Infrastructure.

We used KRSL as the interlingua, and de-
developed the KRIL Kernel, an implementa-
tion of KRSL, as a software layer that translated
forms, queries, and assertions into and out of
the Loom knowledge-representation system.

The Infrastructure TIE also developed the
first implementation of a knowledge server,
an agent that maintains shared knowledge
and information, and that mediates between
planning systems and databases containing
additional information. The knowledge
server comprised the KRSL Kernel, Loom,
the Loom Interface Module (LIM), and the
Intelligent Database Interface (IDI). Loom
was integrated with IDI through LIM, an in-
termediate layer. IDI also communicated
with an external Oracle database containing
most of the relevant data from IFD-2, in-
cluding geographic locations, force descrip-
tions, and so on. The knowledge server sup-
ported external queries and assertions in the
KRSL interlingua, as communicated through
an implementation of the KQML model
(which we'll discuss later in this section).
The server obtained information for its an-
wers either directly from the Loom knowl-
edge base or from the Oracle database via
LIM and IDI.

Figure 2 shows the general architecture of
the knowledge server and communications
substrates that were developed during the In-
frastucture TIE and used later in the CPE. The
TIE communications software was built
around the Paramax (now Unisys) imple-
mentation of KQML, a distributed communi-
cations model for intelligent software agents
that was developed from a specification for the
DARPA Knowledge-Sharing Project.

KQML supports the encapsulation and rout-
ing of messages composed of KRSL (or other
language) expressions. During the TIE, mes-
sages were passed among the knowledge
server and the planning systems Fmerge and
the Prototype Feasibility Estimator (PFE)
(both systems are explained later in Table 3).
(Fmerge and PFE had participated in IFD-2, where communications were by file transfer.)
KQML was implemented as a wrapper
consisting of two pieces of software — a
router and a Knowledge Router Interface Li-
brary (KRIL) — that were added to each
component agent to handle its remote com-
 munications. The router formats expressions
in KQML message syntax and forwards them
through the distributed network. There is one
router per implementation language (for ex-
ample, there is one Common Lisp router).
The KRIL is a representation-language-spe-
cific interface to the router that acts primarily
as a representation translator (for example,
from Loom to KRSL and back) and local dis-


tributes (queries, replies, assertions, and so on)
that the agent will support. In the TIE, the
same Common Lisp router was used by
Fmerge, PFE, and the knowledge server.
Fmerge and the knowledge server used the
KRIL to translate KRSL messages into their
local Loom representations, and PFE used a
special-purpose KRIL that was developed
later into a more general package to support
the translation of KRSL into Common Lisp
Object Specification descriptions.

Communications between agents, which
consisted primarily of representations of mil-
itary plans and deployment schedules, were
standardized as much as possible using what ex-
ist of the evolving shared-domain ontology.

The final demonstration for the Infra-
structure TIE recreated a portion of IFD-2,
this time using KRSL and KQML to access
and store intermediate information and re-
sults in the knowledge server. Using KRSL,
the demonstration asserted a Socap major-
force plan into the knowledge server. When
Fmerge was invoked, it issued KRSL queries
to the knowledge server for the major-force
plan and definitions of the forces involved,
and it received that information as KRSL
forms based on the shared-domain-ontology
definitions of the objects involved. Fmerge
then produced a detailed deployment plan,
and asserted that result back into the knowl-
dge server, from where PFE could retrieve
it for a transportation feasibility analysis.

The infrastructure prototype configuration
was extremely flexible, allowing various
components to reside on the same worksta-
tion or to be distributed across the Internet. In
the demonstrated configuration, Fmerge ran
at Rome Laboratory in upstate New York and
communicated over the Internet with a
knowledge server running at Paramax in
Philadelphia.

We instrumented much of the initial infra-
structure to collect timing information. As
might be expected for a first prototype, many
performance issues were unresolved, and some
steps were quite slow, especially given the vol-
ume of data that was transmitted (for example,
it took 20 minutes to retrieve approximately 1
Mbyte of information about forces for a plan).
This information guided us in the subsequent
development of the CPE Testbed.

The attempted integration of KRSL, LIM,
and Loom also raised representational is-
issues and inconsistencies. One issue was how to
handle the relationship between class-level in-
formation as stored in database records and
the corresponding classes or concepts in the
knowledge base. For example, a data table
might contain information about types of air-
craft (average speed, cargo capacity, and so on)
that might be represented as attributes of
class-level concepts in Loom or KRSL.
Unfortunately, at that time LIM and IDI only sup-
ported the mapping of database records into
instances. Another issue was the mapping be-
tween KRSL objects with subparts and repre-
sentations of that information in the database;
relational databases have difficulty handling
recursive or cyclic relations effectively.

Using external reasoning to support gen-
erative planning. A single, uniform-strategy
AI reasoning tool is not always sufficient to
tackle a real-world problem. In some cases,
legacy software must be integrated with the
AI system, or the AI system must be altered
(for example, with a customized user inter-
face), or additional automated reasoning
methods must be added to supplement the
AI system’s capabilities.

Three TIEs supplemented a mature generative planner (Socap) with several other substantial pieces of AI technology: temporal reasoners, a case-based reasoner, and a constraint-based scheduler. These experiments were among the first that harnessed mature AI systems with completely orthogonal development paths to solve the same problem. They were unique in two ways: they used existing, independently developed AI-based modules to supplement an existing generative planning system, and they added capabilities that were novel to, or relatively unexplored in, generative-planning systems.

Temporal reasoning to support generative planning with SIPE-2. When Socap was originally developed as an application of SIPE-2, SIPE-2’s limited temporal-reasoning capability surfaced as a shortcoming. By itself, SIPE-2 cannot reason about resource use, or place temporal constraints between actions in the plans. Consequently, the generated plans do not represent important constraints in the planning application domain.

SIPE-2 treats time strictly as a consumable resource: time can be consumed but not produced, and its consumption over parallel tasks is nonadditive. Each action specification can have associated start-time and duration variables, but SIPE-2 calculates specific values for time variables only when the constraints force a particular value; otherwise, it computes the allowable range.

SIPE-2 uses three techniques to establish the relative orderings of actions: it inserts ordering links to avoid resource conflicts; it uses one action to meet several other actions’ precondition requirements by ordering those actions in the plan; and it coordinates separate subplans by adding ordering links between subgoals of the two subplans. These techniques allow SIPE-2 to solve many simple temporal problems, but it still cannot represent the time constraints of two possibly unordered actions. Two SIPE-2 actions either are ordered with respect to each other or are unordered. If they are unordered, the planner might order them either way or execute them simultaneously. It is not possible to model when the various effects of an action become true during its execution, or when actions must occur simultaneously.

By adding as a support system a temporal-constraint manager for Allen’s temporal-relations calculus, we can explicitly represent actions starting or finishing at the same time, actions overlapping each other, or one action occurring during another. In this way, Socap can represent many dependencies between different military actions. For instance, cargo off-load teams should arrive at the destination airport or seaport when the first air or sea transport arrives.

TIE 4 extended Socap’s ability to represent and reason about time by adding a layer on top of SIPE-2 that would track the temporal constraints in the plan, initially using the Tachyon temporal reasoning system to maintain and propagate these constraints. The interface to Tachyon is general enough to permit a different temporal reasoner to be substituted, in particular, TMM. (The article on page 10 briefly discusses both Tachyon and TMM.) This integration allowed Socap to represent more sophisticated temporal constraints in plans and to reason more accurately about the times and durations of actions and about resource use over time.

The approach used to integrate temporal reasoning into SIPE-2 was coarse-grained in the sense that complete sets of relations were sent for analysis, rather than sending constraints to be added incrementally to a graph maintained by the temporal reasoning system.

A plan critic, run at the end of each planning level, extracted the temporal information (time windows and intermode constraints) from the plan generated to that point, and sent it to a temporal reasoner. The revised temporal constraints that were returned were stored in the plan’s nodes. We also added methods to maintain these constraints on the plan as Socap expands goals to a new level. We extended SIPE-2’s operator syntax to let designers specify any of the 13 Allen relations or quantitative constraints (the permissible range of metric distances) between the endpoints of any pair of nodes in an operator.

The extended system found temporal inconsistencies in previously generated plans that could be resolved only by changing the dates on which military units were available to perform missions, or by assigning different units to those missions. This shows that the system now reasons with a more complete model of the military-domain constraints. The added temporal information lets Socap pass a more complete and consistent set of constraints to the scheduler.

The temporal reasoning that TIE 4 introduced into SIPE-2 does not provide representation and reasoning support for continuous and interruptable events. However, the limited capabilities that were added provide significant power to deal with important constraints in the military-planning domain, and they improve the generated plans substantially.

Case-based reasoning for force selection. Selecting the right force to participate in a military operation and tailoring a force to meet an operation’s requirements are important parts of operations planning. When selecting or tailoring forces, a planner must consider a unit’s potential to deter or defend against an enemy threat, the unit’s mobilization, its ability to handle the terrain, and its time to deploy.

Originally, when expanding a mission goal, Socap presented a list of all available force units that satisfied the SIPE operator constraints for that type of mission, and asked the user to select a unit. The user could see what constraints were met by the units in the list, but had to rely on personal knowledge of the mission context and of each possible unit’s capabilities to determine the most appropriate unit. Socap could not rank or prioritize some units if all units satisfied the necessary constraints for a job.

We determined that case-based reasoning could provide this kind of preferential force selection and tailoring. TIE 2 addressed this issue by integrating Socap with the Case-Based Force Selection (CAFS) system developed at GE. CAFS indexes and retrieves descriptions of forces from a case library of force units, based on the units’ mission requirements, climate, terrain, mobility, and other related information.

TIE 2 modified Socap to call the CAFS
module for major force selection, instead of presenting a list to the user. CAPS uses the constraints and context information provided by the Socap operator and bindings to find potential force units for a mission (including units that were tailored for past missions) and to rank their appropriateness. If the closest matching force does not fit Socap's requirements, CAPS can apply heuristics to modify the force appropriately. For example, it might be necessary to add support units to the retrieved force structure to account for differences between the prior and current situation.

Constraint-based scheduling as a capacity-analysis tool. Assessing a plan's transportation feasibility (essentially the use of resources over time) is a major concern for military planners. IFD-2 demonstrated why Socap needed to get feedback from external plan-feasibility evaluation tools to produce better, more transportation-feasible plans. To investigate this issue, and to explore ways of overcoming Socap's simplified model of resource management, TIE 3b integrated Socap with a constraint-based scheduler called Distributed Transportation Scheduling in OPIS (Ditops). TIE 3b modified Socap to call Ditops at various stages of Socap's search through the space of possible plans. Ditops assesses the feasibility of the current partial plan, based on transportation-resource capacity requirements. This early analysis helps assign resources to operations, based on projections of resource bottlenecks. Either Socap or a user can use the analysis results to choose feasible deployment destinations for major forces during initial plan generation, or to reassign transportation resources.

To focus Ditops' attention on the relevant parts of the plan, Socap extracts a temporally ordered plan network that includes only the transport operations and other non-resource-using actions that contribute temporal-constraint information. For each operation, Socap supplies the resources (planes, ships) assigned to transport specific units, and supplies the holding capacity required on those resources to transport the personnel and materials involved. Ditops' capacity-analysis routines analyze this information, and Ditops returns the result to Socap in the form of comparative estimates of the expected availability versus demand for aggregate resources, by class of transport, over time. This data appears on a color graph showing expected demand on a resource type versus available capacity over time. Users (and eventually Socap itself) can then reallocate resources or change an operation's time.

Force elaboration and transportation scheduling. The data for a military deployment plan is currently maintained in an operational form called a TPFDD, for time-phased force-deployment data. This data is a table of the units to be deployed by air or sea to specific destinations, their sizes (in tons of air or sea cargo), and the dates they are available for shipment and need to arrive at their destinations. TPFDDs are critical to military-planning operations: they are input to simulators to estimate a plan's transportation feasibility, and they are also input to scheduling algorithms to produce detailed schedules and manifests for shipments of air and sea cargo.

Unfortunately, developing TPFDDs is labor-intensive, and primarily performed by the command staffs of the individual armed services. Once the Joint Staff has developed a general operations plan, the services must fill in the details of the operations and determine the lists of necessary personnel and equipment. They often do this by cutting-and-pasting from detailed force lists developed for prior missions; without the proper software support, this is largely a manual data-entry task. For large operations, this process can involve TPFDDs containing thousands or tens of thousands of entries.

To fill in the required data elements in each TPFDD record, these service planners must create a rough schedule of the deployment, specifying the ports along each unit's deployment path, and the time windows for each unit's arrival and departure from those ports. Frequently, planners do this with only approximate information about the available transportation resources, and even less information about the constraints on loading and unloading cargo at individual ports. They are forced to do this because they cannot specify directly what they know — namely, the ordering and timing constraints that are based on their knowledge of military planning and of the plan under construction (for example, which units need to arrive before other units). If planners could specify this information directly and conveniently, then more dynamic and automated scheduling and rescheduling of the plan's transportation elements would be possible when better information about transportation resources was available.

ARPI has demonstrated how to automate some of these staff functions, given the right information. In IFD-2, BBN and ISX developed Fnerg as a simplified first step toward automatically generating TPFDDs directly from operations plans produced by Socap (Fnerg and Socap later became part of the CPE Testbed). Fnerg approximates parts of the manual process, expanding each force in the Socap plan into a list of units and materials in three ways: by using a force module library, by augmenting the list of operational units to be deployed with a number of other units performing combat support and service support (such as local transportation, medical services, and food services), and by adding units to the deployment plan based on doctrinal rules about resupply of materiel (this last activity is more fully realized in the Loggen system).

Fnerg was a demonstration prototype designed to be replaced by other ARPI technologies. Two TIES have been steps in that direction. TIE 6 improved the ability of CAPS to intelligently adapt configured units to new situations. TIE 3a used Ditops to develop scheduled departure- and arrival-time information for TPFDDs. We'll now look at TIE 3a in more detail.

Constraint-based TPFDD scheduling. Ditops is based on the OPIS job-shop scheduler. In job-shop-scheduling, an item can usually be manufactured in several ways by using sequences of operations, each of which can be accomplished by tools that exist at
Ditops works with largely completed TPFDDs, which specify the goods to move and the destinations, but which can leave other decisions open. Ditops then builds detailed schedules of the ships and planes that will transport each good, combining transportation-feasibility analysis similar to what military simulators had done before, with schedule development. TIE 3a relaxed the constraints on how complete a TPFDD had to be to do this scheduling.

The military rule of thumb for large operations is that 20% of the cargo should move by air, and 80% by sea. However, given timing considerations and available lift options, this heuristic can easily be wrong. By relaxing some of these constraints on the shape of the scheduler’s output, but providing more information about the plan’s structure and dependencies, Ditops can often produce transportation plans and schedules that are more efficient and more easily and automatically revised.

To achieve this efficiency, TIE 3a gave Ditops additional information about the deployment plans for IFD-2. This information replaced the fixed, but estimated, dates for unit movements in a TPFDD. Given this information, Ditops balanced the use of ports and avoided some bottlenecks caused by port-capacity overload. Similarly, by avoiding early commitments on how cargo was to be shipped (air or sea modes), Ditops found efficient transportation for each unit (for example, using excess air transport capacity to improve arrival times), and better balanced the limited resources available.

**The CPE Testbed**

Leading up to the first IFDs and TIEs, ARPI contractors developed and used largely domain-independent AI tools to adapt military planning applications to the specific domain of military transportation planning. After the first round of TIEs, we began to combine these technology components into the CPE Testbed, a single distributed environment based on the interoperability model used in the Infrastructure TIE. (Table 3 briefly describes several of these systems.) We designed the testbed to:

- provide an infrastructure for future TIEs that supports experimental metrics and their evaluation, and uniform access to data and scenarios for transportation planning;
- establish a common representational basis for remote communications between planning and scheduling systems;
- facilitate the transition of technologies into IFDs and operational prototypes; and
- enable the exploration of coordination and control issues in a heterogeneous re-planning and rescheduling system.

We developed the CPE Testbed Release.

---

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEVELOPER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socap</td>
<td>SRI</td>
<td>Socap applies the SIPE-2 generative planner to military employment planning. It consists primarily of a domain model and operators for a range of plans that address variants of the IFD-2 scenario. It also includes tools for scenario data management and for communication with other ARPI components. It has its own user interface.</td>
</tr>
<tr>
<td>CAFS</td>
<td>General Electric R&amp; D Center</td>
<td>CAFS applies case-based reasoning to major-force selection during employment planning. It works with Socap or a human user. It is built on top of Primo and the Caret case-based reasoner, which was developed by GE and is scheduled for release into the Common Prototyping Environment.</td>
</tr>
<tr>
<td>Ditops</td>
<td>Carnegie Mellon Univ, Robotics Inst.</td>
<td>Ditops is a reengineered version of the OPIS constraint-based scheduling system. A constraint set and support tools enable its use in military-deployment transportation scheduling and transportation resource-capacity analysis.</td>
</tr>
<tr>
<td>KTS</td>
<td>Kestrel Inst.</td>
<td>KTS is an efficient constraint-based scheduling system for transportation deployment scheduling. It was developed using Kestrel's KIDS semiautomatic program-generation environment.</td>
</tr>
<tr>
<td>Format</td>
<td>Mitre</td>
<td>Format is a knowledge-acquisition tool for retroactive annotation of force modules. It enables a case-based reasoner (or a person) to index and reuse the modules.</td>
</tr>
<tr>
<td>Loggen</td>
<td>Mitre</td>
<td>Loggen computes the amount of sustainment (food, supplies, and so on) that a military operation will require. Loggen has recently been incorporated into the Dynamic Analysis and Replanning Tool (DART), for use in operational contexts.</td>
</tr>
<tr>
<td>PFE</td>
<td>BBN and Carnegie Mellon Univ.</td>
<td>PFE is a Lisp implementation of a military transportation simulator, modeled after those that the military used in DART to estimate a deployment plan's feasibility, given a set of available transportation resources. PFE's data input and output requirements were a model for both Ditops and KTS. PFE was the first tool developed specifically for the CPE, and is one of the most frequently checked-out tools of the CPE Repository.</td>
</tr>
<tr>
<td>Fmerg</td>
<td>BBN and ISX</td>
<td>Fmerg expands Socap-generated plans into detailed military deployment plans for IFD-2. It is a simplified model of a force-package elaboration and augmentation system. It comprises several subsystems that together span the gap between a high-level description of a set of forces with missions, to be deployed as part of a plan, and the detailed descriptions of those forces, their subcomponents, and various combat support and sustainment elements, all of which must be deployed.</td>
</tr>
</tbody>
</table>
resentations and a translator of SIPE-2 plan
communications that occurred in the TIEs
During that time, we transformed the ad hoc
into consistent KRSL representations, in-
cluding the development of KRSL plan rep-
resentations and a translator of SIPE-2 plan
representations to KRSL forms. We devel-
oped support for remote starting and meter-
ing of software modules, partly by replacing
some of the underpinnings of the prototype
KQML software with Cronus, a more mature
object-oriented communications substrate,
and by adding a simple, uniform interface
(KNET, a simple variant of KQML) to that
substrate. (We used Cronus primarily be-
cause of its relative maturity, its multiple-
language support (including C, C++, and
Lisp), and its tools for remote process control.)

Substantial effort and voluminous e-mail
traffic was devoted to standardizing the com-
munication language between the modules
using KRSL. To standardize the functions of
the different systems for experimental com-
parison, all communications between mod-
ules were organized around standard sets of
messages for pairs of functional classes of
modules. Each system was classified as one
of six types of modules: plan generator, force
selector, temporal reasoner, force expander,
deployment simulator/scheduler, or knowl-
edge server. For each type of module, we de-
efined the KNET messages to which that
module had a contract to respond, and we
characterized inputs and outputs descrip-
tively and in terms of KRSL syntax. Figure
3 shows the interaction paths for which these
communications were defined.

The first version of the CPE Testbed had
one plan generator (Socap), two temporal
reasoners (Tachyon and TMM), one force se-
lector (CAF5), one force expander (Fmerg),
two deployment simulators/schedulers (PFE
and KTS), and one knowledge server, plus a
client system called the CPE Testbed Exper-
imenter’s User Interface, which provided re-
 mote process and experiment control, but
which was not a server that responded to
messages. Ditops was soon added as a third
simulator/scheduler. Recently, the commu-
nications between Ditops and Socap (from
TIE 3b) have also been implemented in the
CPE by giving Ditops a second role as a re-
source capacity analyzer.

Our attempt to standardize the content of
communications between modules was
largely successful, although time constraints
forced some hard choices. In particular, the
communications between plan generators
and the temporal reasoners continued to have
the batch-oriented flavor of the original TIE
interface between Socap and Tachyon, even
though TMM, the other temporal reasoner,
was designed for more incremental interac-
tions with clients. (We are planning an ex-
periment that uses TMM more as it was in-
tended, in conjunction with the O-Plan2
generative planner. This should lead to a re-
vised interaction style for temporal reason-
ers in the CPE.) Similarly, some aspects of
SIPE-2’s plan language that are unique to
SIPE-2 were carried over into KRSL and the
shared-domain ontology. Ironing out these
issues will require further effort by the entire
research community.

Other improvements over the Infrastruc-
ture TIE included improved message encod-
ing that reduced the size of large messages
threelfold, and decreased their transmission
time tenfold. For example, the new encoding
reduced the transmission size of a TPFDD
object from 1.5 Mbyte to 0.5 Mbyte, and re-
duced the transmission time from 20 minutes
to 2 minutes. Additional speedups also oc-
curred when accessing remote databases
from the knowledge server.

The testbed as a platform for technology
evaluation. The CPE Testbed was designed
to facilitate the running of experiments that
test the speed and effectiveness of new tech-
nologies, both singly and in TIE-like combi-
nations. To run an experiment, one chooses
sets of modules that will run in each of a
number of “trials.” A mechanism for apply-
ing software “alligator clips” or meters to dif-
f erent points in the computations was in-
cluded in the CPE and KNET software. By
simple declarations, we can collect informa-
tion about run times and process parameters
trial by trial, where each trial might vary
which module performs a particular prob-
lem-solving function. For instance, simply
by setting up a sequence of trials in the CPE
Testbed Experimenter’s User Interface, we
compared Tachyon and TMM by having
Socap call them with the same sequence of
inputs in successive trials. To evaluate ex-
perimental results, the interface incorporates
the Clasp analytical statistics package from
the University of Massachusetts, so we can
use a variety of statistical measures to di-
rectly analyze the data collected by meters.

Experimenters can also use the interface
to specify the hosts on which the modules
will run during a trial. All active hosts in the
same CPE configuration can be used —
whether they are at BBN in Cambridge, ISX
in Los Angeles, Rome Laboratory in New
York state, or any other developer site around
the country — as long as the modules in-
volved are installed at those hosts.

Current status of the CPE Testbed. Cur-
rently, twelve modules can communicate via
the CPE’s distributed communications
model: the user interface, Socap, Tachyon,
TMM, CAF5, Ditops, KTS, Fmerg, PFE,
Format, the knowledge server, and an exper-
imental plan server for plans developed by
the Theater-Level Analysis, Replanning, and

Figure 3. Defined KRSL communications paths in the first CPE Testbed.
Graphical Execution Toolbox (Target) from IFD-3. These systems represent the contributions of ten different corporations or university laboratories. Several research prototype systems, including 0-Plan2, Services and Information Management for Decision Systems (SIMS), and CoBase, are also under consideration for incorporation into the CPE. Other, more operationally oriented systems under consideration include Loggen, the Air Campaign Planning Tool (ACPT), and the Analysis of Mobility Platform (AMP), a successor to the Dynamic Analysis and Replanning Tool (DART, from IFD-1).

The Common Prototyping Environment has successfully demonstrated the potential for increased collaboration and sharing among a large set of R&D projects. It has validated our methods for promoting more effective software and information sharing, and it has motivated a large community of researchers to increase the interoperability of their work, and to establish experimental metrics for progress.

But this environment can still improve and mature. Some researchers found the CPE less useful than we had hoped, in some cases because it became available after research had already begun, in other cases because of its size and the compromises involved in designing an environment that integrates a wide group of software systems with current technologies. Most of the systems in the tested used AI technologies that predated ARPA. Addressing the needs of the university researchers who are developing the next generation of tools is now one of our biggest concerns.

It is a challenge to devise an architecture for diverse, heterogeneous software systems, all participating in solving complex planning problems. Dealing with the strong human element to these distributed systems increases the challenge. Our goal is not just to build knowledge-intensive distributed software systems; we must build systems in which intelligent planning software assists many people at many different locations in their dynamic, ongoing efforts to coordinate the activities of the US Armed Forces, one of the largest organizations in the world.

References


Mark H. Burstein is a senior research scientist at BBN Systems and Technologies. For the last five years, he has been the technical project leader for the Common Prototyping Environment. His research interests include planning and scheduling technologies, agent-based simulation, machine learning, plausible and logical reasoning, and knowledge representation. He received his PhD in computer science from Yale University in 1984, and his BS in mathematics from the Massachusetts Institute of Technology in 1976. He can be reached at BBN, 10 Moulton St., Cambridge, MA 02138. Internet: burstein@bbn.com.

Richard Schantz is a senior scientist at BBN Systems and Technologies, where his activities focus on distributed systems technology, distributed-computing, technology transfer, and development of large, complex systems from reusable parts. He received his PhD and MS in computer science from the State University of New York at Stony Brook in 1974 and 1970, and his BS in mathematics from New York University in 1968. He can be reached at BBN Systems and Technologies, 10 Moulton St., Cambridge, MA 02138; Internet: schantz@bbn.com.

Marie Bienkowski directs SRI International's Applied AI Technology program, which transfers basic research and advanced development products into operational prototypes. She was project manager for Scopac. Her research interests include integrating planning and scheduling systems, explanation production for simulations and training, and environments for community knowledge creation. She received her PhD in computer science from the University of Connecticut in 1986. She can be reached at SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025; Internet: bienk@sri.com.

Stephen F. Smith is a senior research scientist and the director of the Intelligent Coordination and Logistics Laboratory in the Robotics Institute at Carnegie Mellon University, where his research focuses on intelligent systems technologies for large-scale planning and scheduling. His research interests include constraint-based reasoning, mixed-initiative problem solving, configurable planning and scheduling systems, and machine learning. He received his PhD and MS in computer science from the University of Pittsburgh in 1980 and 1977, and his BS in mathematics from Westminster College in 1975. He can be reached at the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213; Internet: sfs@cs.cmu.edu.