CaFé: A Group Process to Rationalize Technologies in Hybrid AAMAS Systems

H. Van Dyke Parunak, Marcus Huber, Randolph Jones, Michael Quist, and Jack Zaientz

Soar Technology, Inc., USA
3600 Green Court, Suite 600
Ann Arbor, MI 48105
{van.parunak,marc.huber,rjones,quist,jzaientz}@soartech.com

Abstract. Most agent research seeks insights about a single technology, and problems are chosen to allow this focus. In contrast, many real-world applications do not lend themselves to a single technology, but require multiple tools. In an applied AI company, each tool often has its own advocate, whose specialized knowledge may lead her to overestimate her tool's contribution and diminish that of other tools. To form an effective team, the various members must have a shared understanding of how their tools complement one another. This paper describes CaFé ("**Ca**ses-**Fe**atures"), a group process that we have prototyped for building a consensus mapping between tools and real-world problems. The five AI technologies encompassed in our prototype are cognitive architectures, intelligent user interfaces, classic multi-agent system paradigms, statistics and machine learning, and swarming. Structured group discussion identifies the dimensions of a feature space in which the technologies are distinct. The scheme that emerged from our exercise does not pretend to be an exhaustive characterization of the techniques, but it is a jointly owned map of our technology capabilities that has proven useful in design of new use cases.

1 Introduction

A recurring topic at AAMAS is how to move the results of research into real-world applications. Our company, Soar Technology (SoarTech), provides applied AI solutions to a range of customers. We find that real applications often do not align well with disciplinary boundaries that guide basic research.

Research progress requires focusing the researcher's attention on a particular approach, tool, or technology, so that it can be characterized theoretically, implemented elegantly, and examined with a thorough experimental design.[1] In this setting, it is appropriate to choose problems that are tailored to the features of the being studied.

Customers in the real world usually do not start with a particular method they wish to exercise. Their pressing problems do not respect the convenient categories according to which we structure research and train students. As a result, organizations that

[1] For our purposes, we use the terms "approach," "tool," and "technology" interchangeably.

address real-world needs often assemble a toolbox of capabilities. In our case, we started with a single flagship technology (the Soar cognitive architecture [13]), but over the years have recruited a staff with capabilities quite different from our original focus. In the process, we have encountered a challenge.

Our researchers understand their own approaches very well, and tend to view every problem through a perspective that is appropriate to their own tools. Companies like SoarTech often dissolve into disjoint "centers of excellence," each focused on a single technology, and each marketing to customer problems that align more or less with a center's capabilities. Such a structure under-serves customers in two ways.

First, it may not fully address the needs of the problems to which it does respond. It is not uncommon for a multi-disciplinary company to end up competing with itself on some opportunities, when different technologists want to bring different tools to bear. In such cases, the different facets of the problem might be more thoroughly and robustly addressed if multiple tools could be applied in tandem.

Second, some large and gnarly problems are too complex for a single technical perspective, even for the most optimistic advocate of a single technology. Such problems are typically left to large "system integrators" who may not bring the depth of technical understanding offered by expert researchers. In overcoming the narrowness of academic researchers, system integrators often fall victim to technical shallowness.

As a company, we seek to avoid both the narrow stove-piping of the academy and the shallow technical depth of a large integrator. We want our technical experts to share an understanding of our set of technologies that will enable them to deploy the full strength of their capabilities in synergy with one another. This paper reports on the form and initial results of a group process that we have implemented for this purpose. We expect it to be of value to the AAMAS community in two ways.

First, as a contribution to the software engineering of agent-based systems, it offers a process to enable multi-disciplinary teams to address complex problems that require the hybridization of multiple agent technologies.

Second, though preliminary, the joint feature space that we derived in our initial deployment of the CaFé method may be of interest in its own right.

Section 2 outlines the CaFé process, which draws its name from two information artifacts contributed by each technical advocate: a *Case study* of a problem that is particularly appropriate for her technology, and a list of *Features* of problems for which her technology is appropriate. Section 3 summarizes the specific Cases and Features in our prototype exercise of the methodology. Section 4 reports on the case discussions that form the heart of the process. Section 5 describes the feature space that results from our process. Section 6 demonstrates the use of this feature space in a series of new design patterns. Section 7 offers a concluding discussion.

2 The CaFé Process and Its Context

The CaFé process (Section 2.1) contributes to numerous areas within software engineering (Section 2.2), and brings some discipline to the *de facto* integration of different technologies that other researchers have already identified (Section 2.3).

2.1 Description of the Process

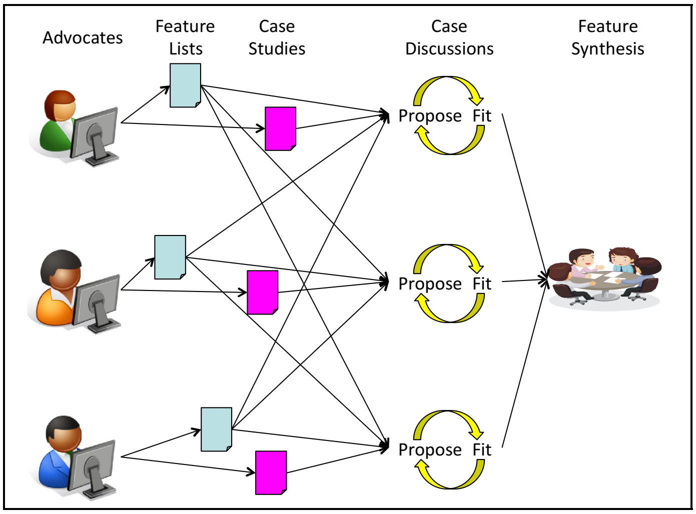

Fig. 1. The CaFé Process

CaFé is a structured group process among advocates for different technologies that encourages them to compare their approaches in the context of several example applications, and helps them to generalize these comparisons as a set of features that make a problem (or part of a problem) appropriate for one or another tool (Fig. 1). Each technology or tool is represented by an advocate who is expert in its use. Each advocate produces two artifacts representing her technology: a *feature list* describing the characteristics of a problem that would recommend the use of her technology, and a *use case* or example problem that she would consider an ideal candidate for deploying her technology. The process of preparing these artifacts before the group begins interaction encourages each advocate to recognize that her technology is better suited to some problems than to others, and to articulate what those problems might look like.

The entire group of advocates then discusses each use case. The discussion includes *proposals* by each advocate of how each technology might contribute to the case, and *fitting* the different technologies into an overall pattern based on the case.

Finally, after discussing all of the individual use cases, the advocates review the features from the individual cases and seek an overall *synthesis* that discriminates among the individual approaches.

The features that result from this process are not as detailed as those initially proposed by the advocates. They do not characterize each technology by itself, but situate it with respect to the other technologies. Most important, they are jointly owned by the advocates as a group, and so can guide collaborative design on new projects.

2.2 CaFé and Conventional Software Engineering

Software engineering is a large and complex discipline, and we view CaFé as a complement to traditional tools rather than a replacement for any of them. To situate the reader, we comment on how CaFé is related to each of the major thrusts of software engineering, as defined in SWEBOK 3.0 [2].

Software Requirements: The various features proposed by technology advocates resemble the characteristics commonly elicited as requirements for a software system (e.g., need for rapid reactivity; support for distributed decentralized operation). However, we found that these characteristics are not sufficient to distinguish the technologies from one another, since the requirements supported by different technologies often overlap.

Software Design: The case discussions typically take the form of proposing high-level designs for the case under discussion, mapping out a high-level architecture for a system to address the needs of the case and nominating the most appropriate technology for each component. From this perspective, CaFé can be viewed as a tool for high-level software design. In fact, the joint feature space that we derived from our case discussions (Section 5) functions as a high-level guide for outlining the architecture of a new system (as illustrated in the examples of Section 6).

Software Construction: Each technology has its own techniques and processes for software construction, which we did not seek to integrate.

Software Testing: Our prototype does not include evaluation, but we discuss possible directions for evaluation in Section 7, and any such process would draw on standard practice in software testing.

Software Maintenance: Good practice in software maintenance cuts across all of our technologies, and we did not explore the contribution of our technologies to it. However, see discussion of "Software Quality" below.

Software Configuration Management: All of our technologies draw on the same supporting systems for configuration management.

Software Engineering Management: We view CaFé as an important contribution to software project management, particularly in the design phase, enabling the integration of insights from different stakeholders.

Software Engineering Process: CaFé is a particular software engineering process that is most valuable in the design phase of a project.

Software Engineering Models and Methods: Each of our technologies has its own distinctive models and methods. We did not explore the interaction among these in this prototype.

Software Quality: ISO/IEC 25010 [12] defines eight product quality characteristics for software (functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability, and portability), of which CISQ has selected four (reliabililty, performance efficiency, security, and maintainability) that its members ranked as most important [3]. One delegate to EMAS 2014 suggested that these characteristics might provide an alternative set of features with which to distinguish technologies, but it is questionable whether one of our technologies is intrinsically more reliable (respectively, efficient, secure, or maintainable) than another. Quality attributes and our features are related at a deeper level: *in the context of a given* *application*, a feature may contribute to one or another quality attribute, and these relations could be identified through an analysis such as the house of quality [9]. This exercise would be a natural and useful extension of our prototype.

Software Engineering Professional Practice: The accepted professional standards for software engineers cut across all of our technologies.

Software Engineering Economics: As discussed in "Software Quality" (above), in the context of a specific application, the choice of technology will make a difference in the economic viability of a solution. Our exercise suggests that heterogeneous designs, which combine different technologies in a single application, will often be more competitive than designs that draw on a single technology throughout.

Computing Foundations, Mathematical Foundations, Engineering Foundations: Each of our technologies draws on distinctive foundations. We did not explore the relations among these in this prototype.

2.3 Other Work in Hybrid Architectures

CaFé is not the first effort to address the question of combining different technologies to solve a single application problem. We have given examples of such hybrids from our own work before [16]. Others have also suggested such approaches. To name only a few: Ferguson's TouringMachines architecture [7] showed the benefits of layering reaction, planning, and modeling horizontally in a single agent, InteRRaP [8] demonstrated vertical layering of different reasoning modalities, and a combination of neural and cognitive methods was the best performer for event recognition in the DARPA Minds' Eye program [4].

Examples of such combinations are valuable as existence proofs showing that hybrid systems are feasible. These examples demonstrate that fundamentally different reasoning modalities and agent architectures can be interfaced with one another. But they were generated *ad hoc*, and provide little guidance to developers seeking to find appropriate hybrid approaches to other problems. CaFé seeks to offer a disciplined approach to hybrid systems. Building on insight from past experiences ([14,16]), it offers a process for designing hybrid systems from the ground up.

3 The Artifacts

We considered five technologies, all familiar to the AAMAS community, in our initial foray with CaFé. Each has a strong advocate on SoarTech's current technical staff, some of whose publications in each area are referenced below.

- Cognitive Architectures (CA) are reasoning frameworks, such as Soar [13,22] or ACT-R [1], that are derived from high-level cognitive models of human reasoning and problem solving, and are intended to produce realistic human-like results. For example, the Soar cognitive architecture explicitly models different facets of human memory (procedural, semantic, episodic) and learning mechanisms

(reinforcement learning, chunking, experience) motivated by experimental results in cognitive psychology
- Intelligent User Interfaces (IUI) are technologies intended to mediate between human users and machine reasoners (e.g., [21,23]). Like cognitive architectures, they are inspired by insights from experimental psychology, but in this case the focus is on insights into the functioning of the human perceptual system rather than internal reasoning mechanisms.
- Multi-Agent Systems (MAS) is a collection of conventional MAS techniques that focus on inter-agent coordination, including BDI models, joint intention theory, theories of trust and norms, and agent communication languages (e.g., [10,11]). These methods are largely inspired by sociological models.
- Statistics and Machine Learning (SML) uses formal statistical methods to characterize data and detect patterns [17,19]. These techniques include cluster analysis, probabilistic graphical models (such as Bayesian belief networks, hidden Markov models, and Markov networks), neural and kernel-based methods, and generative models such as Latent Dirichlet Analysis, as well as a range of techniques for combining multiple statistical methods.
- Swarming harnesses self-organizing methods inspired by natural systems, with many simple agents interacting locally in a shared environment ("stigmergy") [15], usually through scalar fields over the environment that they both generate and sense. Drawing on insights from statistical physics and complexity theory, these methods can yield system-level behavior that is qualitatively more complex than the behavior of the individual agents, a phenomenon known as "emergent behavior."

For each of these approaches, we summarize the features and the case study proposed by its advocate. The purpose of these summaries is not to attempt a definitive statement of each approach, but to illustrate the flavor and level of detail involved in these artifacts. While these descriptions are abbreviations of the documents prepared by our advocates, each of those documents is still only one or two pages long.

3.1 Cognitive Architectures (CA)

Feature List: Cognitive architectures fit problems with these characteristics:

- Multiple simultaneous, interleaving tasks that frustrate the development of linear procedural code, but can be managed by pattern recognition
- Ability to handle and categorize special cases with pattern-driven processing
- Need to execute in real time (not much slower, but also not much faster), using least commitment to support rapid computation of an acceptable answer that can be refined if time is available
- Need for rapid reactivity to changed circumstances
- Need to support explanation of behavior to human stakeholders
- Real-time learning as the agent executes in the domain.

They are a poor choice for problems that involve

- Rapid processing of large amounts of data (more than 10k items per second)
- Sequential batch processing
- Number crunching
- Execution much faster than real time (as in constructive forecasting)
- Offline learning

Case Study: CA would be a good choice for a chef's decision-support assistant. Recipes are declarative representations of "how to cook" something. But having a great cookbook doesn't make someone a great chef. A great chef has extensive procedural knowledge and the ability to substitute, adapt, and handle interruptions and opportunities. Recipes are inherently serial, but cooking a meal requires opportunistic parallelism. A complete system would require situation interpretation and human-system interaction. The chef domain reflects the need for learning in a number of ways.

- Recipes are forms of declarative knowledge.
- Recipes can be taught/demonstrated.
- There is also "book knowledge" about ingredients, cooking techniques, etc.
- Recipes can be generalized and decomposed in goal-based fashion.
- Chefs acquire expertise by practicing cooking.
- Chefs learn about substitutions, short cuts, and handling unexpected events.
- Cooking knowledge can be "recomposed" to create new recipes and techniques.
- Chefs need to communicate with fellow chefs, servers, and suppliers.

3.2 Intelligent User Interfaces (IUI)

Feature List: Systems for which development of an IUI is appropriate tend to have one or more of the following features:

- Human-centric: Humans need to control, understand, and trust the system and its outputs.
- Incorporate human knowledge: The operator (or operators) have knowledge, including long term domain knowledge and short term situation awareness, that can improve system performance and/or outputs.
- Incorporate human decision-making: The operator(s) can make detections or decisions beyond the system's capability or authorization.
- Adaptive / Mixed Initiative: The system needs to adjust its operating characteristics to take into account changing operator (or actor) beliefs, desires, and intentions, both between and within system execution cycles; alternatively, the system needs to prompt the operator (or actor) to adjust their behavior.
- Representation boundaries: The system needs to mediate between two or more frames (typically, a user representation such as a doctrinal air traffic control grammar and a software engineered representation such as an AI planner structure).
- Naturalistic (multi-modal) usage environment: The system needs to interpret multiple streams of user input (mouse, voice, text, pointing) and/or coordinate multiple streams of output (video, audio, haptic).

- Supporting human constraints: The system needs to act for the user in a domain that exceeds human scale (either long time intervals, large data sets, fast reaction time) or that exceeds the specific operator's ability to act effectively (e.g. expert support for novice users, problems of high complexity or very high cost of error).
- Personalization: The system should be tuned to the specific preferences of a particular user or user group (or actor/actor group).

Case Study: It quickly became apparent that any realistic system we discussed would need to interact with human stakeholders, and in the end we did not consider a separate case for IUI, since we were comfortable that the cases proposed by other advocates would serve well to explore its complementarity with the other technologies.

3.3 Multi-Agent Systems (MAS)

AAMAS is accustomed to a broad use of the acronym "MAS" as including any system (including, for example, a swarming system) with many interacting agents. For our purposes, we focused on coarse-grained MAS techniques that rely on symbolic representations. The advocate for this area is expert in agent communication languages, joint intention theory, and related high-level coordination techniques.

Feature List: Problems that are suggest the need for multi-agent systems exhibit some of the following features.

- Teaming: More than one agent is required to solve a problem.
- Distributed: Computational solution needs to be divided (e.g., complexity, location, incomplete information, role, function, computational space/power).
- Synergistic: Using multiple agents gives a better solution that using a single one.
- Robustness: Reduces/removes single point of failure.
- Decentralized: Advantageous for distinct agents to make independent local decisions, processing (e.g. parallelism), or actions.
- Asynchronous: computation and interaction aren't tightly coupled.
- Organization: Structure (interaction, control) between agents important and/or advantageous (e.g., societal, problem structure, communications requirement).
- Heterogeneous: Distinct agents with differing capabilities.
- Dynamic teaming: Components (agents) motivated but not required to coordinate.
- Competitive: agents can work against each other.
- Flexibility: Independent contributors to portions of distributed solution.
- Complexity/Scalability: Multiple agents with localized modeling and reasoning can address larger problems.
- Semantic: Disparate localized representations and meanings.
- Perspective: Modeling and interpreting other components behavior/state.
- Opacity/Compartmentalized: Certain aspects of solution need to be hidden.

Case Study: An MAS approach would be ideal for a mixed team of soldiers and heterogeneous robots. The robots could include ground, air, surface, and subsurface vehicles, each with potentially different types of sensors, effectors, communication modes, and levels of local computation. Special attention needs to be paid to the changing roles of each entity in the team. Communications are dynamic, because of adversarial jamming, complex terrain that limits propagation, and the need for tight coordination. Relations among the units change constantly as the mission unfolds.

3.4 Statistics and Machine Learning (SML)

Feature list: Problems that are suitable for SML exhibit some of these features:

- The availability of large amounts of sensor data (video/audio capture, etc.) to yield useful levels of significance;
- Difficult to reduce data down to a manageable amount of symbolic information, whether because
  - the correct feature set is not known and must be discovered,
  - the data is intrinsically complex (e.g., speech data), or
  - different symbolic reductions are appropriate in different contexts;
- The availability of clear metrics for correctness of data handling to guide learning;
- Training and testing data available or easy to generate at will;
- Black-box with correct output is sufficient; no requirement to explain the interpretation of the raw data;
- Need to handle uncertain inputs, or to produce multiple results with varying levels of numerical confidence

Case Study: Consider the problem of commanding one or many autonomous (or partially autonomous) assets using multiple modalities in a naturalistic way. Such a system would need to integrate speech recognition, gesture recognition (whether visual or by smartphone or smartwatch with gyro and accelerometer), and sketching, as well as traditional computer or mobile device UIs. For user acceptance, the system would need to match existing protocols. For example, in a military context, gestures should be those already used to command infantry, and structured speech forms such as the SALUTE report [6] or the nine-line brief [5] should be followed, so that a mix of human and robotic assets can be commanded simultaneously.

3.5 Swarming (SW)

Feature List: The advocate for swarming characterized appropriate problems as

- consisting of *discrete* parts, such as robotic platforms, people, units of information, or events; if the natural decomposition of a problem is functional or assertional, rather than in terms of a set of entities, another technology may be preferred;
- consisting of *diverse* entities, performing diverse functions, and dealing with diverse information sources (since individual agents can preserve distinctions that would be lost in the mean-field approach of many equation-based formalisms);
- favoring *distribution* of computation across multiple platforms, whether because of communication limits that hinder centralizing data, or because of the need to parallelize computation in combinatorially large problems.
- allowing *decentralized* decision making by individual members of the swarm, within bounds established by the operator;

- subject to *deprivation* of computational resources, since swarming coordination through shared scalar fields is less demanding than symbolic manipulations;
- subject to rapid *dynamic* change that requires constant self-reorganization.

Case Study: SoarTech has several projects in autonomous systems, such as ground robots and UAVs, and our sponsors are interested in assessing the trustworthiness of their autonomy software. Conventional assessments of the trustworthiness of an engineered system are based on statistical analysis of a fault tree describing the structure of the system [20]. Once we endow a system with autonomy, we must also consider different trajectories through mission space and the demands they put on various platform subsystems. We have developed a representation of an extended fault tree that combines a conventional fault tree of the platform with a hierarchical task network representing mission space, but the resulting structure is too complex to explore exhaustively. We propose using swarming agents to compute a probability distribution over alternative mission instantiations, and thus compute the probability of mission failure, analogous to the Top Undesirable Event in a conventional fault tree analysis.

3.6 An Observation

These feature lists and case study nominations were prepared by the advocates independently of one another. Not surprisingly, they are difficult to compare directly. Some of the features do not distinguish between technologies (for example, the ability to respond to dynamic changes in the world). Others have no counterparts across approaches that would allow direct comparison.

This incommensurability of features is not surprising. In fact, it reflects the challenge of designing a hybrid AAMAS system, starting just with a set of technologies. The trade-offs among them emerge only when we consider them in the context of specific problems, motivating the series of case study discussions that we conducted.

4 Case Discussions

After advocates have circulated feature lists, we discuss each proposed case study. As suggested in Section 3, each discussion has two phases (though in our experience the thread of conversation often switches multiple times between the phases). In the proposal phase, advocates suggest how their technologies could be applied to the case, or to extensions of it that might realistically be required. In the fitting phase, the group seeks to fit the various technologies into the specific use case, exploring how to rationalize the role of each technology. This rationalization frequently draws from the feature lists originally prepared by the advocates, but instead of being unilaterally proposed by the advocates, it is the result of a group consensus. Each of these phases yields important insights about the relations among the technologies.

Each case was suggested by an advocate as ideally suited to one specific technology, but the *proposal* phase of each discussion never lacked for contributions from other advocates. As different advocates envisioned how their tools could be applied to a case, the problem tended to expand in scope. Sometimes different tools addressed the same facet of the problem from a different perspective, but more often the viewpoint prompted by a given tool encouraged us to consider a richer, more complex version of the use case, one that looked less like a toy laboratory problem and more like a real-world system. This experience reflects the insight about real-world problems that motivated CaFé in the first place. We hypothesized that such problems would benefit by synergy among multiple approaches, and in fact the more approaches we considered alongside a problem, the more realistic the problem itself became.

In the *fitting* phase of the discussions, we tried to rationalize the complementary contributions of each technology to the (sometimes expanded) case. This rationalization usually took the form of identifying some feature that distinguished alternative technologies *in the context of the case under discussion.* Sometimes these features were already articulated in the feature lists submitted by the advocates in advance, but often they became clear only through discussion of a concrete case.

For example, <<insert case discussion>>

A central insight resulting from our work on CaFé is the difficulty of comparing technologies directly with one another, and the relative ease of comparing them in the context of a specific problem. The individual features lists often claim the same problem characteristics for different technologies, but discussion of a concrete example serves as a catalyst to highlight the differences that matter among the various approaches. Of course, different cases may yield different points of comparison among technologies, but in practice, after we had gone through three cases, we began to see recurring problem features that repeatedly distinguished between tools. We summarized these features in the final feature synthesis discussion (right-hand side of Fig. 1) to define the joint feature space discussed in the next section.

5 The Joint Feature Space

Two dimensions distinguish four of our technologies: CA, MAS, SML, and SW. These dimensions are a) high and low data integration, and b) high and low decomposability (the face of Fig. 2, and Table 1). We were unable to localize IUI in this space in a way that would distinguish it from the other four. Recall that one motive for CaFé is to understand what portions of a complex problem we should address with which technology. To achieve this objective, we seek a joint feature space that distinguishes all of our technologies. To meet this criterion for IUI, we propose a third dimension, c) high vs. low human involvement (Table 2). Fig. 2 shows the resulting overall feature space. This joint feature space is not a definitive characterization of any of our methods, but instead focuses on features that distinguish them from each other.

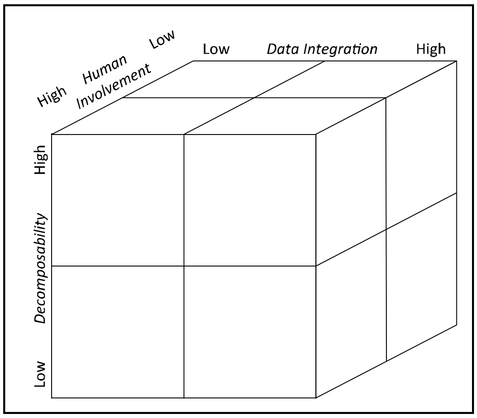

Fig. 2. Joint Feature Space resulting from our execution of the CaFé process

5.1 Data Integration

The Data Integration dimension reflects the degree of linkage among the data items that the problem presents. High data linkage corresponds to a knowledge-rich domain, in which information includes a representation of the semantic relations among data items. In a domain with low data linkage, the relationships among data items are yet to be discovered. Often, problems with low data linkages present a larger amount of data ("data rich" problems), while the knowledge captured in spaces at the high end of the dimension allows the system to work with smaller amounts of data. From a systems perspective, the low integration, data-rich end of the dimension is associated with sensors that access the world directly, while the high integration, knowledge-rich end deals with analysis of data that has been subject to a fair amount of preprocessing. Some aspects of this dimension correspond to the JDL Data Fusion hierarchy [18], in which Level 0 deals with raw signal data, Level 1 identifies objects, Level 2 detects situations among multiple objects, and Level 3 identifies threats.

MAS and CA rely on symbolic knowledge representations, and so are most naturally applied to knowledge-rich problems. SW and SML can use data without such a knowledge overlay and suggest relations among data items that might later be represented explicitly. They can use a knowledge structure as a template against which to compare raw data (for example, using SML with a symbolic grammar), but they do not require this knowledge to be embedded in the data at the outset.

Several of the features suggested by the advocates for individual approaches align with this dimension.

- CA identified the need to explain its reasoning to humans, which requires high semantic content in its representations.
- SML recognized that it is most appropriate when the problem needs a "black box" solver that cannot explain itself.
- SW's use of scalar fields to support deprived applications reflects its focus on data with low semantic integration.

However, by themselves these independent features are not nearly as useful in deconflicting the technologies as is the data integration dimension that emerged as we discussed the application of these tools to common problems.

5.2 Decomposability

The decomposability dimension reflects the degree to which the problem invites solution by multiple interacting components. For problems with high decomposability, it is natural to distribute the solution process across multiple platforms. The most natural processing approach for problems with low decomposability presumes that all information is available on a single platform.

Where the data integration dimension grouped MAS and CA against SML and SW, the decomposability dimension groups MAS and SW against CA and SML. Both MAS and SW use multiple computational entities, but differ in how they coordinate these entities: the stigmergic coordination common with SW agents is subsymbolic, relying on scalar fields over the environment, while MAS agents exchange symbolic information. But in both cases, the information available to individual agents is limited, and differs from agent to agent. CA and SML assume low decomposability. Most examples of CA assume a monolithic reasoner (like the human whose cognition these architectures are intended to imitate). While some clever methods for distributing SML computations have been explored, the fundamental model on which SML rests is the development of a single joint distribution over the variables of interest, which can then be marginalized as required, a computation that is most readily done with all the data in one place.

Again, this dimension reflects some features identified initially by tool advocates:

- SW is applicable to distributed, decentralized problems.
- MAS similarly recognized Teaming, Decentralized, and Distributed as problem characteristics that favor its application.

The case discussion, unlike the individual feature lists, showed the need for low decomposability for most effective application of SML and CA.

These two dimensions effectively distinguish four of our approaches (Table 1). However, IUI did not fit neatly into this taxonomy, leading to a third dimension.

Table 1. Feature Space (without IUI)

		Data Integration	
		Low (Data-Rich)	High (Knowledge-Rich)
Decom-posability	High (multiple agents)	Swarming	Multi-Agent Systems
	Low (single agents)	Statistics & Machine Learning	Cognitive Architectures

5.3 Human Involvement

By definition, IUI technologies facilitate interaction of a human user with an automated system. One can envision a system drawing on our other approaches that does not interact with a human (for example, a closed-loop control system). But when the system as a whole requires human involvement, a user interface is required, and increasingly these interfaces use some degree of AI to facilitate the interaction. So we distinguish IUI from the other four technologies along a "Human Involvement" dimension on which the others are low and IUI is high.

Though IUI is applicable across the entire space spanned by the two dimensions of Table 1, it takes different forms in different areas of this space, depending on the other processes with whi-ch it interacts, as shown in Table 2.

Table 2. IUI Variants for High Human Involvement

		Data Integration	
		Low (Data-Rich)	High (Knowledge-Rich)
Decom-posability	High (multiple agents)	Data Visualizer	Peer Decision-Maker
	Low (single agents)		Cognitive State Inspector

- In data-rich domains, IUI predominantly supports data retrieval and visualization. It allows humans to guide automated reasoners (whether SW or SML) (for example, by identifying information requirements, or presenting knowledge templates to which data should be fit), and it presents to the user the structures discovered by underlying SW or SML processing. It naturally supports an interactive approach to data exploration
- In knowledge-rich, highly decomposable domains, IUI naturally allows humans to function as peers alongside computational agents. IUI presents the user with information that is sent to her from other agents, and translates human input into messages that are exchanged with other agents.
- In knowledge-rich domains with low decomposability, IUI enables the user to interact with a single CA agent (e.g., to inspect or modify the agent's state).

The Human Interaction dimension directly reflects the multiple references to people in the original IUI feature list, including "Human centric," "Incorporate human knowledge," and "Incorporate human decision-making."

6 Some New Design Schemata

One of our motives in developing CaFé was facilitating the design of systems to address large, complex problems that require synergy among multiple AI approaches. In this section, we sketch a series of design patterns that illustrate the value of the feature space that we have developed. We could simply present hybrid designs for the case studies that we discussed, but to demonstrate the extensibility of our results, we instead describe a series of concepts distinct from the original case studies, but drawing on the same joint feature space.

It is legitimate to ask how feasible it is to tie these different methods together in a single architecture, as these designs suggest. While we have not explored interface mechanisms explicitly in CaFé, our experience, shared by others who have built previous hybrid systems (Section 2.3), is that interfacing components based on different technologies is a matter of engineering rather than a major hurdle requiring research.

6.1 Data Fusion and Shared Situational Assessment

A common problem in many domains, both military and industrial, is gathering data from many sensors monitoring the physical world, discovering patterns to develop a knowledge-rich characterization of the current situation, and then assuring that all decision-makers share a common view of that situation. Fig. 3 shows how our technologies might interact in such a system.

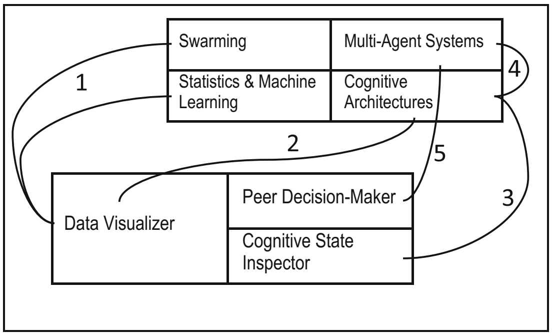

Fig. 3. Schema for Data Fusion and Shared SA

1. Both SW and SML deal with the raw data and detect regularities and patterns, which they expose to a human through a data visualizer IUI. The human in turn can guide the SW and SML agents to refine her view of the world, and refine and enhance the structures that are discovered.
2. Enriched with explicit knowledge through the actions of the human operating the data visualizer, the data can now be consumed by a CA agent that reasons over it in the light of other knowledge (including previous states of the world, mission plans and objectives, and hypotheses). The CA agent can also identify linkages that the human should further explore through the data visualizer IUI.
3. A cognitive state inspector IUI allows the human to monitor the reasoning of the CA agent and perhaps adjust it.
4. The CA agent shares its conclusions with other agents via MAS interfaces, achieving shared situational assessment across the team.
5. Some of these agents may be humans, who participate in the team via a peer decision-maker IUI.

We intentionally leave the links between components in this and the following schemata undirected. In general, we believe that information will flow in both directions; a more refined design would distinguish the nature of the flows in each direction.

6.2 Complex Pattern Detection in Data

Modern data analytics faces a tension between data that are too atomic to be diagnostic and knowledge that is too complex to guide search. For example, a single negative Tweet about US policy might be an isolated comment, part of an emerging viral propaganda campaign, or motivation for an invitation to a public demonstration. These alternatives require different responses, and detecting them depends on patterns involving multiple Tweets. Yet traditional methods of matching an overall pattern against high-volume, high-velocity data do not scale with the complexity of the pattern, particularly if the pattern encompasses several alternative possibilities, only one of which may match. Such patterns are too complex for efficient single-item queries, but the processing to match complete patterns is combinatorially infeasible.

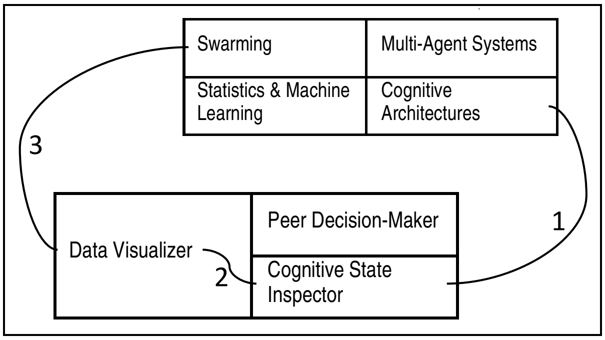

Fig. 4. Schema for Complex Pattern Detection

We are developing an approach to such problems that fits the schema in Fig. 4.

1. A major challenge in knowledge-based systems is authoring the knowledge that drives the system. Currently, complex queries are assembled manually, but our schema anticipates the role of a CA agent in helping a human develop these

patterns, perhaps on the basis of learning from past experience (not shown in the figure). A cognitive state inspector IUI facilitates this interaction.
2. This link indicates interaction between two different human roles: the pattern author (via a cognitive state inspector IUI) and the person using the pattern to interact with the data (via a data visualizer IUI). These may be the same person, or different specialists.
3. To avoid the combinatorial complexity of matching the entire pattern at once to massive data, we use swarming to evaluate the probability that different portions of the pattern are supported by the data, then estimate the value of alternative atomic queries in sharpening these distributions, and execute those queries, all under the supervision of a human via a data visualizer IUI.

6.3 Multi-unit Combat Simulator

A major application area for MAS is in constructive combat simulations. Fig. 5 shows a schema that supports the development of a simulator for a multi-component force.

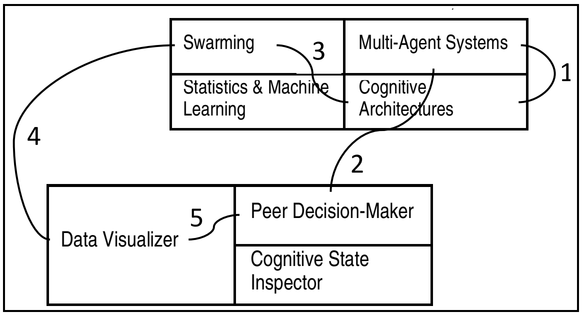

Fig. 5. Schema for Multi-Unit Combat Simulator

1. The simulator's core is a set of CA agents, interacting through MAS interfaces.
2. The MAS organization allows humans to participate in the simulation via a peer decision-maker IUI, realizing the increasingly popular LVC (Live-Virtual-Constructive) mode of simulation.
3. One important feature of cognitive reasoning is anticipating future events. CA agents include some mechanisms for anticipation, but in anticipating geospatial motions, swarming has proven to be a powerful tool.
4. Human players can also benefit from the anticipatory view provided by swarming, via a data visualizer IUI.
5. The data visualizer and peer decision-maker IUIs in this case may be integrated to support a single human player.

6.4 Model Fitting

A recent project gathered opinions from humans via a (non-intelligent) interface to fit weights to knowledge models that let us estimate the similarity behind the human judgments informing the elicited opinions. Fig. 6 shows an expanded version of this system.

1. A CA agent, directed by a human via a cognitive state inspector IUI, develops the knowledge model that is to be fitted to the elicited opinions.

2. Swarming over the model develops the weights on individual edges in the model.
3. The differences between the spectra of weights from different informants are evaluated statistically.
4. The resulting measures of informant similarity then enable a CA agent (which may or may not be the same one involved in the original model authoring) to make more intelligent use of the opinions elicited from the different informants.

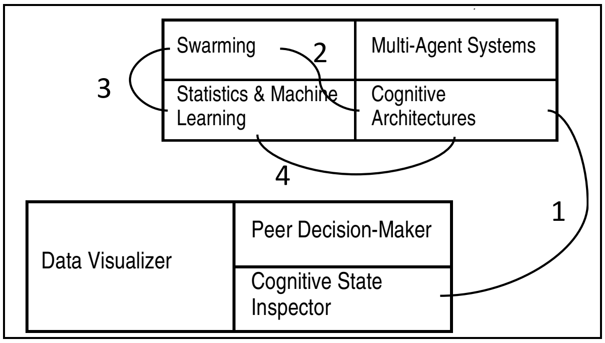

Fig. 6. Schema for Model Fitting

7 Discussion and Conclusion

The method described in this paper enabled experts in different AI specialties to develop a shared feature space showing how their tools complement each other. In turn, this feature space was effective in initial design of new systems beyond the case studies that drove the CaFé process itself.

Our exercise was a prototype of CaFé. We discuss its extensibility and alignment, and how this technique might be evaluated.

By *extensibility*, we mean the behavior of the feature space as new technologies are added to the collection, and as we consider new problems.

We begin with extensibility to new technologies. The five we considered in this exercise do not by any means exhaust the repertoire that we have currently in house, not to mention others that we may acquire. One can imagine game theory in its many variations, distributed constraint optimization, and logic programming, to name only a few. Will adding others require redoing the whole process, yielding a feature space that is radically different from what we discovered for our initial five approaches?

Our experience with IUI is evidence that we can expand the feature space incrementally rather than having to redo it each time we add new technologies with new advocates. IUI did not fit cleanly into the two-dimensional space that the other four approaches suggested. However, adding the Human Involvement dimension allows us to disambiguate it from the other approaches, and careful attention to the nature of the original two-dimensional space allows us to tease apart different techniques within IUI that do exploit the insights of the two-dimensional space.

A related aspect of extensibility concerns the robustness of the joint feature space as we consider new problems. We developed the design schemata in Section 6 to test whether the feature space could be applied to problems other than those that stimulated its definition in our case discussions, and the results encourage us that the space is in fact robust across a wide range of problems.

By *alignment*, we call attention to the fairly minimal overlap between the original feature lists submitted by the advocates, and the dimensions of the resulting feature space. Because the Human Interaction dimension was introduced to distinguish IUI from the other approaches, it is not surprising that this dimension corresponds very closely to the features enumerated by the IUI advocate. However, other individual feature lists include a great deal of information and insight about individual approaches that is not captured explicitly in the dimensions of the joint space.

Some details of the original feature lists do align with the dimensions of the joint space. In addition, this observation about alignment reminds us again of the distinctive purpose of the joint space. Unlike the original feature lists, it is not intended to define each technology, but rather to show how they complement each other. Unused features in the original lists are a reservoir on which we may draw as we consider new technologies and new problems, to refine our understanding, not of technologies in isolation, but of the joint technical space that we are positioned to exploit.

An important but complex question is how one might *evaluate* CaFé. Framing such an evaluation would require identifying a) competing approaches, and b) some figure of merit. Conceptually, software quality attributes [3,12] provide a disciplined approach to measuring the merit of a finished system, but in spite of the existence of numerous hybrid systems (Section 2.3), we know of no other methodology with which one might compare CaFé. We hope that by exhibiting one approach to the problem, we will stimulate others to suggest modifications or competing approaches, that eventually could support a disciplined evaluation. For now, the performance of CaFé can only be evaluated by comparing its products with systems whose technical composition is driven by the informal politics of the developing organization.

Perhaps the most powerful insight from the CaFé experience is the ability of concrete problems to facilitate comparison of different technologies. The usefulness of a third object for clarifying mappings between two other objects suggests that a category theoretic model might be a useful way to formalize the CaFé process and lead to automated tools to support it, a direction that we hope to pursue in future work.

References

[1] Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychological Review 111(4), 1036–1060 (2004)

[2] Bourque, P., Fairley, R.E. (eds.): SWEBOK 3.0: Guide to the Software Engineering Body of Knowledge, 3rd edn. IEEE, Piscataway (2014)

[3] CISQ: CISQ Specifications for Automated Quality Characteristic Measures. Object Management Group (2012), http://it-cisq.org/wp-content/uploads/2012/09/CISQ-Specification-for-Automated-Quality-Characteristic-Measures.pdf

[4] de Penning, L., d'Avila Garcez, A.S., Lamb, L.C., Meyer, J.-J.C.: Neural-Symbolic Cognitive Agents: Architecture, Theory and Application. In: Lomuscio, A., Scerri, P. (eds.) The 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), pp. 1621–1622. IFAAMAS, Paris (2014)

[5] Department of Defense: JP 3-09.3, Close Air Support. Washington, DC, Department of Defense (2009)

[6] Department of the Army: FM 2-22.3 (FM 34-52), Human Intelligence Collector Operations. Washington, DC, Department of the Army (2006)

[7] Ferguson, I.A.: Touring Machines: Autonomous Agents with Attitudes. Computer 25(5), 51–55 (1992)

[8] Fischer, K., Muller, J.P., Pischel, M.: InteRRaP: Unifying Control in a Layered Agent Architecture. German Research Center for Artificial Intelligence, Saarbrucken (1995), `http://www.dfki.uni-sb.de/~pischel/interrap.html`

[9] Hauser, R., Clausing, D.: The House of Quality. Harvard Business Review 66, 63–73 (1988)

[10] Huber, M.J., Kumar, S., Lisse, S.A., McGee, D.: Integrating Authority, Deontics, and Deontics and Communications within a Joint Intention Framework. In: Huhns, M., Shehory, O. (eds.) The 2007 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007). IFAAMAS, Honolulu (2007)

[11] Huber, M.J., Kumar, S., McGee, D.: Toward a Suite of Performatives based upon Joint Intention Theory. In: The AAMAS 2004 Workshop on Agent Communication (AC 2004), New York, NY (2004)

[12] ISO: ISO/IEC 25010:2011: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models ISO (2011)

[13] Laird, J.E.: The Soar Cognitive Architecture. MIT Press, Cambridge (2012)

[14] Lesser, V., Corkill, D.: Challenges for Multi-Agent Coordination Theory Based on Empirical Observations. In: Lomuscio, A., Scerri, P. (eds.) The 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), pp. 1157–1160. IFAAMAS, Paris (2014)

[15] Parunak, H.V.D.: 'Go to the Ant': Engineering Principles from Natural Agent Systems. Annals of Operations Research 75, 69–101 (1997)

[16] Van Dyke Parunak, H., Nielsen, P., Brueckner, S., Alonso, R.: Hybrid Multi-agent Systems: Integrating Swarming and BDI Agents. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS (LNAI), vol. 4335, pp. 1–14. Springer, Heidelberg (2007)

[17] Quist, M., Yona, G.: A novel robust algorithm for structure-preserving embedding of metric and nonmetric spaces. Journal of Machine Learning Research 5, 399–430 (2004)

[18] Steinberg, A.N., Bowman, C.L.: Revisions to the JDL Data Fusion Model. In: Hall, D.L., Llinas, J. (eds.) Handbook of Multisensor Data Fusion, pp. 2.1–2.19. CRC Press, Boca Raton (2001)

[19] Taylor, G., Quist, M., Hicken, A.: Acquiring Agent-based Models of Conflict from Event Data. In: IJCAI 2009. AAAI Press, Pasadena (2009)

[20] Vesely, W., Stamatelatos, M., Dugan, J., Fragola, J., Minarick, J., Railsback III, J.: Fault Tree Handbook with Aerospace Applications. NASA, Washington, DC (2002), `http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf`

[21] Wood, S.D., Zaientz, J.D., Beard, J., Fredriksen, R., Huber, M.: An Intelligent Interface-Agent Framework for Robotic Command and Control. In: The 2004 Command and Control Research and Technology Symposium, San Diego, CA (2004)

[22] Wray, R.E., Jones, R.M.: An introduction to Soar as an agent architecture. In: Sun, R. (ed.) Cognition and Multi-agent Interaction: From Cognitive Modeling to Social Simulation, pp. 53–78. Cambridge University Press, Cambridge (2005)

[23] Zaientz, J.D., Beard, J.: Using Knowledge-Based Interface Design Techniques to Support Visual Analytics. In: Workshop on Intelligent User Interfaces for Intelligence Analysis at IUI 2006, Sydney, Australia (2006)