The Design and Development of a Cloud-based Platform Supporting Team-oriented Evidence-based Reasoning: SWARM Systems Paper

–blinded–

Abstract

The Smartly-assembled Wiki-style Argument Marshalling project (SWARM) commenced in 2017 as part of the US Intelligence Advanced Research Projects Activity (IARPA) funded Crowdsourcing Evidence, Argumentation, Thinking and Evaluation (CREATE) Program. The SWARM project has developed an online platform allowing groups to produce evidence-based reasoning. This paper provides a summary of the core requirements and rationale that have driven the SWARM platform implementation. We present the technical architecture and associated design implementation. We also introduce core capabilities that have been introduced to encourage user interaction and social acceptance of the platform by the crowds.

1. Introduction

The Internet is awash with data and opinions. This data is increasing exponentially and will likely continue to do so for the foreseeable future [1]. There have been enormous advances in the ability to harness and utilize such data through advanced computational techniques and access to large scale infrastructures leveraging for example high performance computing and Cloud resources. The term "big data" has now entered mainstream vernacular as the ability to derive understanding and knowledge from this data deluge [2]. This is often through advanced data mining, machine learning and information retrieval involving multiple terabytes of data. However whilst the technologies around big data processing have seen fundamental advances in computational abilities, the ability for reasoning and understanding has not significantly progressed.

The IARPA funded CREATE program seeks to address this issue. Specifically, the CREATE program focuses on development and evaluation of platforms that use crowdsourcing and structured analytic techniques to improve analytical reasoning, with specific focus on the demands of the Intelligence Community. The CREATE program commenced in 2017. It involves four funded projects that are developing systems that are targeted specifically at supporting improvements in evidence-based reasoning. The four teams include: *TRACE – Trackable Reasoning and Analysis for Collaboration and Evaluation* which focuses on the development of a web-based application that uses crowdsourcing to overcome common shortcomings in intelligence work by improving the division of labor and reducing both the systematic and random errors individuals generate while promoting communication and interaction among teams; *Co-Arg – Cogent Argumentation System with Crowd Elicitation* which focuses on the development of a software-based cognitive assistant for intelligence analysts that tests hypotheses, evaluates evidence, sorts facts from deception and provides intelligent reasoning about potentially rapidly evolving situations; *BARD – Bayesian Argumentation via Delphi* which uses causal Bayesian networks as the underlying structured representations of argument analysis and augments this with automated Delphi methods to bring groups of analysts to a consensus-based analysis, and *SWARM – Smartly-assembled Wiki-style Argument Marshalling* which focuses on the development of a user-oriented, web and Cloud-based platform that supports crowd-based reasoning with specific focus on supporting end users and achieving improved, collective reasoning.

The CREATE program itself represents a 4.5-year effort comprising three phases. Phase 1 of the program (January 2017 - September 2018) focuses on the development of the core platforms and their evaluation by independent teams of users. The nature of the questions (problems) posed on Phase 1 are typically smaller-scale constrained problems, i.e. where all of the information is provided in the problems that are set. The projects are now entering the final part of Phase 1. The ultimate aim of the platforms in Phase 1 is to clearly demonstrate that they provide a

significant improvement over other approaches, e.g. use of Googledocs and Google Hangouts for collaborative editing and communication. Those CREATE platforms that meet these criteria will proceed to Phase 2 where the problems that are posed are expected to be more complex and potentially include unconstrained problems, e.g. involving external resources and data.

This paper focuses on the rationale that has driven the design and development of SWARM.

2. Related Work

There has been an extensive body of research undertaken into improving evidence-based reasoning with specific focus on the needs and demands of the Intelligence Community [3]. [4] identifies a range of techniques that can be used to tackle issues faced by intelligence analysts in their daily reasoning activities. These include approaches to address challenge judgments, identify mental mindsets, stimulate creativity and manage uncertainty. Whilst [4] identifies a range of techniques that can be used by analysts, there is a dearth of actual evidence that any given structured technique actually improves improve on-the-job reasoning. Whilst some techniques have been shown to promote good thinking in other contexts, e.g. there is evidence that argument mapping can improve critical thinking skills [5], the adoption of any specific tool or approach by the intelligence community has not materialized. There are many potential explanations for this. The suitability of any given tool for the given problem at hand; the demands required to master particular tools; the fact that tools may not be suited to the particular workflow or process that reflects the actual daily needs of analysts; the requirements for tools that can be general purpose and used for different scenarios depending on the problem at hand. For many analysts, the workflow around their reasoning and collaboration is primarily based upon drafting of individual documents, e.g. Word documents, and iterating with peers/experts for feedback and comments. Such an approach does not lend itself to improvements in reasoning nor benefit from the any/all experts (people).

There are however many examples of systems that have evolved and become *de facto* places for expert opinions and answers [6]. These are not based upon any given expert or analyst, but through garnering collective expertise through crowdsourcing [7] [8]. Wikipedia is one of the major web-based platforms that has leveraged the global community [9]. Whilst not specifically focused on reasoning, it is clear that Wikipedia has established resources that have shaped global knowledge based on the wisdom and knowledge of (global) crowds. The wisdom of crowds has been studied by numerous researchers [10] [11] [7], yet the translation into mainstream software solutions that can be adopted directly by the Intelligence Community has failed to occur. Whilst platforms such as Wikipedia are focused largely around factual information, they do not directly lend themselves to the typical problem-based requirements facing the intelligence community.

Platforms such as StackOverflow (www.stackoverflow.com) provide many features that are highly desirable when leveraging the wisdom of the crowd and more closely aligned with the needs of the Intelligence Community. Within StackOverflow, any user can post any technical question to the platform, e.g. how to tackle a specific software problem. It is noted that StackOverflow has now been rolled out and supports many other disciplines. Following a given posting, any user can post a potential answer to the question and all users are able to comment and vote on each answer. Depending on the rating of all users, the best (most highly-rated) answer floats to the top and in that sense constitutes the crowds answer to the question. This approach is highly successful and is heavily used by many software developers globally as the place to have software issues tackled. This depends greatly on the reputation system that reflects author contributions and ensures that the community self-monitor any potential gamification of the system, e.g. having users rate each other to increase their ranking on the platform [**?**] [12]. For many technical developers, the ranking of an individual on StackOverflow can be used as both a token of esteem but also to directly aid career development. There are many desirable features from a platform for tackling potentially arbitrary questions from the community at large. The fact that the solutions have been adopted by mainstream software developers is testament to their success. Such recognition has driven the core ideas that have driven the SWARM user requirements, however there are some significant differences which we enumerate.

3. SWARM Requirements

The SWARM platform has been developed through a highly agile software development methodology. The technical team has implemented numerous versions of the platform that have been iteratively developed with feedback and ideas shaped continually by the domain scientists and experimental teams working on SWARM. The earliest prototype focused largely on argument mapping that allowed for users to tag their inputs as hypotheses, evidence, arguments for/against,

assumptions etc. This version of the platform was delivered within the first few months of the project. It was rapidly identified that this approach had major issues in how teams perform and interact. The project thus pivoted in the core approach taken. Ultimately, it was recognised that the success of SWARM would be based on development and delivery of a platform that encourages users to engage and not on forcing them down an argument mapping solution. The platform should support their activities, encourage team work and minimise the technological demands that too often place a hurdle in adoption of a given technology. The SWARM platform was designed to meet a specific range of criteria including:

Contribute reports and collectively select the best report. SWARM users should initially draft their proposed responses on a given problem independently of any other users' work. The individual drafting process should allow users to get their thoughts in order and develop their argument before opening their work up for review. Once a user publishes their response to a given question, other users can comment and propose improvements to it. This is an iterative team-based process that underpins SWARM. At the end of lifecycle to a given problem that is set, the highest-rated report is selected to be the team's submission. Users should also be able to contribute snippets (resources), that aid the team's work, e.g. diagrams or analysis, that may not form part of the final report directly but serve as an aid to improving the team's efforts.

Feedback and review. SWARM's rating system needs to be designed so that users can give fine-grained feedback to the authors of any given contribution. This helps contributors figure out which aspects of their reports need more work. Users can also comment directly on reports, or discuss them informally via chat. These features encourage users to improve their work by incorporating suggestions and feedback. Users should not be able to rate their own contributions on the platform.

Engagement. SWARM needs to be designed to support users who are solving difficult and often dry, fictitious problems. Given this, boosting participant engagement and motivation is a major design requirement. As such, the SWARM platform is required to make the user experience as smooth as possible. This should include a clean and intuitive user interface and solutions that encourage engagement at any time.

Social warmth. For team based activities, SWARM recognised that there is a clear need to support social interactions between users to foster social warmth. Social interaction motivates users to engage. Chat and comments both promote social interaction. Users should be able to tag each other in chat and send/receive notifications. If they wish to share credit when another user has provided helpful feedback, they can nominate other users as co-authors of their reports. Sharing credit promotes the spirit of collaboration.

Few constraints on users. SWARM should not be not designed to replace or automate human reasoning. It should not ask users to use any particular structured technique, which might constrain them to only solve a subset of possible problems or dissuade them from engaging due to the effort required to learn to use any given tool. Rather, SWARM should be designed to support, encourage and motivate people to work hard on a wide range of problems. Users should be able to contribute their work in the format that they are most comfortable with. SWARM should provide users with a rich text editing environment, supporting the ability to drag and drop images and allow them to incorporate work from potentially many diverse tools.

Access to support for structured analytic techniques and other problem-solving tools. While SWARM users should not be constrained to use any particular technique, they should be supported in the use of a variety of different techniques. The need for a wide array of "lenses", i.e. ways of viewing and approaching problems, should be offered. Users should be encouraged to use the tool that best fits the problem at hand without any mandate to adopt a given solution. The SWARM platform should incorporate some core supporting tools, but provide a rich compendium (LensKit) of tools that the users might find beneficial. This should include basic training and education.

Anonymity. Users should be provided with pseudonyms and avatars to allow them to express opinions more freely, and to rate each others' work without being prejudiced by their prior opinion of the author. Ratings should also be kept anonymous; users can see how many other users rated their work, but cannot see who did so, allowing users to rate without fear of retaliation.

Support large team sizes. SWARM should be designed for potentially large crowds with no adverse impact on the performance of the platform. Through the container-based approach that has been taken to deliver and deploy the components, the system has been designed to scale horizontally across the Cloud [13]. This leverages Docker technology and the Kubernetes technology [14].

4. SWARM Architecture

The SWARM ecosystem has been built around the concept of micro-services as opposed to development

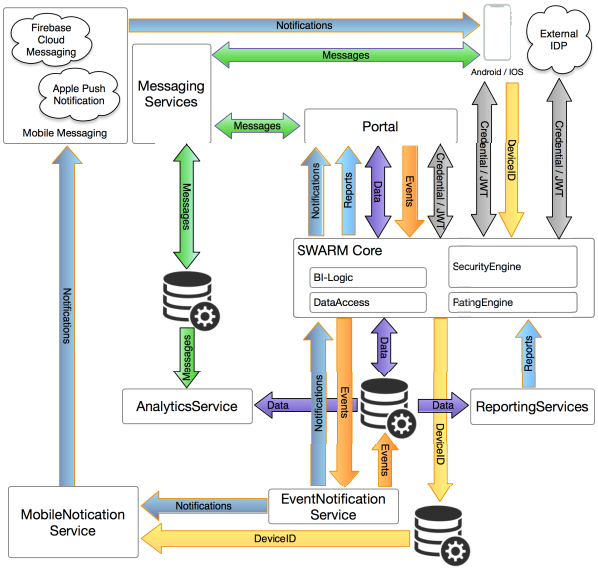

Figure 1. Overall SWARM System Architecture

of a single monolithic application. Such an approach allows to containerize the system components and hence promote system scalability. This is especially important with large numbers of users. This lends itself to Cloud infrastructures. In this section we identify the key SWARM components and their associated functionality. Figure 1 presents a high-level overview of the (current) SWARM system architecture.

As shown in Figure 1, SWARM is comprised of a number of key components. We enumerate these and outline their core functionality in the following sections.

The *SWARM Portal* is the main component that supports user interactions and the user experience as a whole on the SWARM platform. Users are grouped into teams and each group can have its own set of problems. Given this, there is a clear need for authentication and authorisation capabilities in the platform. Users can discuss the posted problems with their peers as well as construct their own response to the problems that are posted. As well as posting their own responses, users can comment and rate other team member responses. The Portal provides access to a range of other tools that have been developed. This includes an Argument Mapping / Graphical Tool, access to a Lens kit for concepts and techniques that can help and guide users to reason and thus produce quality responses, as well as capabilities for probabilistic reasoning. All users are assigned an Australian-oriented animal-based pseudonym on the platform, e.g. Dingo47 as well as an associated avatar. Users are discouraged from disclosing their actual identities to avoid any offline discussions. It is expected that all interactions and communications occur on the platform.

Given the ubiquity of mobile devices for accessing web content, SWARM also provides users with native mobile applications (Android® and IOS® - including both iPhone and iPad applications). These mobile apps offer similar functionality as the Portal, but in a manner that is tailoured to the limitations of the screen space of mobile devices. The development and delivery of mobile applications encourages engagement in the platform, e.g. users can access the content and be notified of updates or chat with other users at any time.

The *SWARM Core* provides the main subsystem of the SWARM ecosystem. This subsystem is comprised of several key components including the *Business Intelligence Logic, Data Access Security Engine* and a *Rating Engine*. The *SWARM Business Intelligence Logic* component defines and executes the business rules that shape the behaviour of the SWARM platform. There are a number of business rules governing the user interaction within SWARM. A given problem lifecycle typically consists of a number of states. When a problem is posted by an administrator, it can be in one of three states: *Frozen*, *Open* or *Closed*. When a problem is in the *Frozen* state, users can only read and construct their private responses, i.e. their responses cannot be made public until the state of the problem is changed to *Open* by the problem owner (the administrator). A problem does not remain active/open forever. Typically problems are open for a certain period of time in which the users/analysts are encouraged to produce interact with one another to produce responses. Once the problem state is changed to *Closed*, users are unable to add responses or comments to the problem. Users can, however still view closed problems. Events and notifications that are activated within SWARM are also governed by a set of business rules that are defined within the business intelligence component. The granularity and frequency of these events and notifications are fully configurable.

The *SWARM Data Access* component provides the data persistent layer of the system. This component is responsible for marshalling/unmarshalling of objects prior to transmission to/from the database service. This component provides a separation between the objects and the actual process of storing/retrieving data. Such a layer provides a degree of transparency over the underlying database and hence reduces the coupling between the system logic and need for data serialization.

The *SWARM Security Engine* is responsible for access control, authentication and authorization. The *Security Engine* authenticates users through a local database and third party identity providers such as Google®. For external identity providers, the OAuth2

Figure 2. SWARM Rating Score

[15] framework is used. Once authenticated, users are checked with regard to their associated permissions, e.g. to determine which group the user belongs too as well as their role. Users with admin roles have higher privileges compared to ordinary users. Such admin roles are able to publish new problems and define the duration of problems for instance. Other users are typically only able to analyse, rate, chat and otherwise contribute to the production to the responses to problems that are set within their own team.

The *SWARM Rating Engine* supports and manages the evaluation aspects of user contributions. It provides users with the ability to (privately) rate other user responses to the given problem. Various forms of rating such as simple vote (like) or scale-based (0-100) are supported by the engine on different dimensions of the responses as shown in Figure 2. More complex and detailed evaluation-based ratings leveraging more flexible, problem-related rubrics are also supported. The rubric can provide user-insights on various issues of reasoning and argumentation. The *Rating Engine* also provides a means to reconcile and aggregate all user ratings for all responses to a particular problem. It supports wide-ranging modes of aggregation from simple or weighted averages (for all ratings related to a specific response) to complex matrix-based aggregates (across all users and all responses within a problem). The result of this aggregation is used to select the best response for the problem of interest. The *SWARM Rating Engine* utilizes a number of strategies to rate and score responses including: *BandRating*, *RatingScore*, and *RatingCount*. Here *BandRating* classifies responses based on the number of ratings a response receives. A response will be classified in a higher band if it receives a certain threshold value. The *RatingCount* on the other hand is given as the total number of ratings a response receives. The *RatingScore* is an overall score given to a response.

The *SWARM Messaging* subsystem is the backbone of the communication (messaging) functionality available in SWARM. With this functionality users are able to discuss various aspects online via an online chat facility. As a crowd-based application, this messaging service compliments SWARM as it promotes the interaction between users in the crowd. The messaging service is available to both the Portal and mobile applications. The SWARM messaging service is an XMPP (Extensible Messaging and Presence Protocol)-based application server which enables near real-time message exchange between client devices over the network.

Users in SWARM are grouped into teams, thus the *Messaging Service* must also provide services that are aligned with such grouping. Users from a given group should not be able to see discussions/chats occurring in different groups. To achieve this, the *Messaging Service* relies on the authorization service provided by the *Security Engine*. When a user logs into the SWARM system, the access token provided to the *Security Engine* contains authorization information in its payload. Using such information, the *Messaging Service* is able to control the message visibility and hence preserve the privacy of groups.

The *SWARM Reporting Service* is charged with generating the final report. As a crowd-based reasoning platform, SWARM does not depend on an individual to produce the best report. Instead, it automatically selects the best response as contributed to, and rated by, the team to produce the final report. The process of selection is done by sorting responses based on their *BandRating*, *RatingScore*, *RatingCount* and the final *TimeStamp*. Responses with a higher *BandRating* will be selected first regardless of whether their *RatingScore* is lower than the *BandRating*. Such a strategy is used because it is believed that a response that has been lowly rated by hundreds of people (i.e. with a higher *BandRating*) is likely to be more accurate and realistic compared to a report that has only been rated by a single individual even though its *RatingScore* is high. If there are more than one response with an equally high *BandRating* then the process of selection is continued. In this second stage, responses with the highest *RatingScore* will be selected. If that still does not produced a single result, then the response with the highest *RatingCount* will be selected. Finally, if the previous process of elimination does not produce a result then the time of the update to a response will be used as the basis for deciding on the final report that is actually submitted.

The *SWARM Event Notification Service* supports the capture and preservation of events that have occurred in the system. Capturing and preserving events is

important because it allows data analytics to be applied to gain insight from the individual and team user behaviour when interacting with the system. This also allows for identification of features requiring further development in the system to increase the overall user experience of SWARM.

Events in SWARM are typically generated by the core subsystem which directly interacts with users via the Portal. Thus events represent user activities. For certain user activities, e.g. when a user places a comment to a response or creates a response, an event is triggered (generated). The events generated are sent to the *Event Notification Service* where they can be preserved. The *Event Notification Service* also produces notifications on certain events. These notifications are sent to the relevant users either via the web browser or via the mobile applications using push notification technology. For example, when a user post comments to another user's response, the latter will be notified immediately. Similarly, if a response was posted, the members of the related group will be notified. This encourages such members to analyze the responses and rate them accordingly. In order to accommodate portal-based users that are not online, the *Event Notification Service* buffers notifications that could not be delivered. Re-delivery will be attempted when the target users are online and reconnect to the SWARM portal. The *Event Notification Service* maintains a list of user browsers used to connect to SWARM. When there is a notification required to be sent to a particular user, a simple look up extracts the list of the browsers where the user is activated. The notification is then sent to the user. The *Mobile Notification Service* complements the *Event Notification Service*. Its main task is to manage the notification delivery service required for mobile devices. Unlike the *Event Notification Service* which maintains a list of browsers associated with users, this subsystem maintains the unique IDs of mobile user devices. Thus a single notification can be sent to multiple devices belonging to a single user. When a mobile application is used for the first time (after installation), it connects to *SWARM Core* and registers itself by sending the device ID of the mobile application to the core. This device ID is then persisted in the database. Upon receiving a request from the *Event Notification Service* subsystem to send a notification to a user, this component performs a simple look-up to retrieve a list of devices associated with that user. The information returned also contains the type of devices, either iOS or Android-based. Appropriate mobile messages are then constructed accordingly according to the device type. These messages are finally set to either Google's FCM (Firebase Cloud Messaging) or Apple's APN (Apple Push Notification) platforms for final delivery to the mobile devices. It should be noted though that the *Mobile Notification Service* does not buffer notifications that failed during delivery, rather it relies on the service of FCM and APN to redeliver the notifications.

The *SWARM Analytics* subsystem provides insights about user behaviour within SWARM based on the analysis of user activities. This can include creating responses; usage of argument mapping tools; use of the lens kit; comments posted to other team member responses; events; message contents; usage of web and mobile applications as well as any demographic information. By analysing the interaction between users in the platform and in the chat/messaging facility, the SWARM platform is able to identify which users are the most active users in a group and how teams interact more generally. Such information can be used to tackle attrition, e.g. for users that have not engaged for some time, notifications can be sent to encourage them to re-engage.

All of the above components and services are deployed as containers using Docker containerization technologies. This approach allows SWARM to be easily deployed and scaled horizontally across Cloud infrastructures. Currently SWARM has been deployed on two different cloud infrastructure: the National eResearch Collaboration Tools and Resources (NeCTAR - www.nectar.org.au) Research Cloud - the national cloud for academics/researchers in Australia, and to Amazon Web Services (AWS - www.aws.amazon.com).

5. SWARM Realization

The SWARM system has been designed to support groups of people working collaboratively on activities related to evidence-based reasoning. The typical workflow is that a problem is defined and released to multiple separate groups to work on. SWARM embraces the fact that each individual in a group will have their own approach, style or preferred approach to support high quality reasoning. SWARM allows various work styles. For example, users can discuss a problem in the chat forum or by submitting a proposed response immediately. Supporting the idea of contending analyses, users are also encouraged to try and compare different analytic approaches. To support this, SWARM provides a SWARM Lens kit, where a logical lens can be seen as any kind of tool, concept, pitfall, checklist or technique that can help structure or guide users' thinking. This compendium of techniques is available for users, but there is no direct expectation

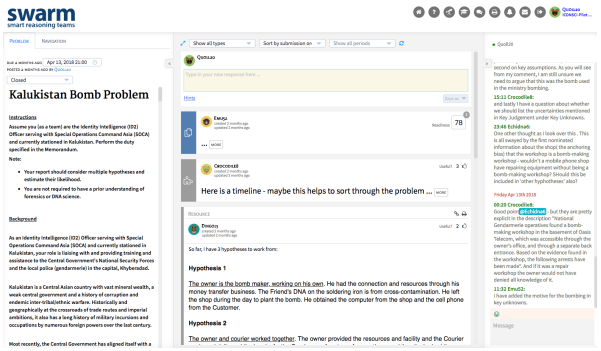

Figure 3. SWARM Portal

that users must adopt any particular lens.

Figure 3 shows the main screen of the SWARM portal after the user has navigated to a particular problem. The screen provides a multi-panel interface. The left side of the screen shows the problem. This is typically always accessible to users when they are working on (responding to) a given problem that has been set. The middle panel shows the group responses and comments and ranking related to the responses. The right side shows the chat panel. This split panel approach enables a seamless workflow where the user can draft a response while consulting the problem description or discuss the problem in the chat with collaborators at the same time. Depending on what the user focuses on, the panels can be enlarged or minimized as desired.

The middle panel of Figure 3 is the main work area. Here users can submit responses, either as a fully fleshed out answers (reports) to a problem or as useful *resources*, where a resource is anything that could be helpful to the team, e.g. an evidence table, a figure or a list of alternative hypotheses. The panel itself is currently based on a simple web-based editor (CKedit - www.ckedit.com) that supports text entry and formatting. This has been extended in various ways to meet the editing, graphical and analytical needs of SWARM.

A report is typically a more structured, formal answer which covers various aspects of the problem. Users are able to work on their contributions in isolation in a private, draft state (indicated by a yellow background). When they feel that they are ready for scrutiny by their group members, they can publish them. Since SWARM has been designed to cope with large crowds, the system provides features to expand, collapse and filter the information in the middle pane. Responses can be sorted in a variety of ways: by report submission order, by average rating, by the creation date, by recent changes or based on the author. User can react to responses, either by commenting on them or by rating them. Resources and comments can be rated by their usefulness with a simple thumbs up/thumbs down. Reports can be rated in detail as shown in Figure 2 based on their readiness to become the final report for the team. The readiness rating includes aspects of reasoning such as the completeness, correctness, logic, evidence or alternatives used, as well as aspects of communication such as clarity or format. Ultimately the rating of the reports that are generated determines which report is to be used as the basis for the team's final report. It is noted that teams will typically identify the best report at a given point in time before the closing date of the problem and iteratively work on refinements and improvements to this single report before the final submission deadline.

By default the left side panel shows the problem description along with general information about its due date and its current state (frozen, open or closed). There can be many questions that are listed, both open and closed. Selection of a given question will automatically open the second navigation tab that shows a list of responses and their ratings. On the right side of Figure 3 is the chat panel. This shows the discussions occurring amongst the group members giving them the chance to exchange thoughts about the problem in a more spontaneous manner. Users can also address other group members by tagging them. It is important to note that the chat panel encourages informality, e.g. the chat room can be used for general discussions that are potentially outside the remit of the problem focus. This encourages social warmth and collaboration.

It is noted that the portal front end has undergone a range of experiments and evaluations with regards to its user interface design. This has included video recording and monitoring of individuals in how they might use the platform – with specific focus on those users that have never seen or heard of SWARM, and users that are not savvy with web-based systems. This has directly shaped the user experience aspects of the platform including the layout, content and naming conventions used. The current portal version reflects the general consensus from the users involved in these experiments on what the system should offer to be as user-friendly as possible. More complicated features of the platform are suppressed and only available after opening up sub-menus. A key yardstick of the platform is attempting to ensure that the training requirements are minimal.

In addition to the core features of the portal, the SWARM platform also provides several targeted tools including a *Probability Calculator* and a *Graphical Tool*. As noted, the SWARM design philosophy

encourages a diversity of analytical approaches to solve problems. This is a deliberate choice to encourage participation. The idea is that autonomy in analytical approach will make users feel less constrained and therefore more willing to engage with the system. However, there are tools and approaches that are especially suited for certain kinds of problems. For example, in situations where problems deal with uncertainties, it is useful to have tools for dealing with probability calculations. The design principle of ”latitude” in analytical approaches, versus the need for specifically relevant tools led to the idea of a ”non-intruding” simple tool for probability calculations. That is, rather than constraining users to approach analysis of uncertainty in a particular way, SWARM allows users to use any tool, but also makes available a simple ”probability calculator”. The probability calculator is embedded into the text editor and can be accessed by simply typing a probability formula inside ”backticks”. A ”calculate” button is available on the editor toolbar to make the system evaluate the probability formula and render a result. Formulas may be written in structured English, or in an abbreviated mathematical form. For example, the ”probability of Rain is 20%” is acceptable, as is its abbreviated form of ”pr Rain=0.2”. A probabilistic problem may be formulated with probability assignment statements like these, as well as conditional probabilities like the ”chance of SprinklerOn given Rain is 1%”. Once the problem has been formulated, calculations can be made like the ”probability of WetGrass given Rain” as shown in Figure 4 (left).

In essence, the probability calculator can infer probabilities from simple Bayes networks, represented as structured English formulas. In this way, the platform provides users with a simple (and optional) tool to reason about probabilities. The tool attempts as much as possible to shield the user from the technicalities of the probability calculations themselves. The design principle reflects the kind of approach inherent in pocket calculators, whereby a user need not know how a ”square root” or ”compound interest” is calculated, but merely how such calculations are used, and what parameters are needed.

Technically, the calculator uses a simulation approach to solving collections of probability formulas. That is, each probability variable is represented as a bit array that is randomly assigned in proportion to the given probability for that variable. Relational calculations such as the ”probability of X given Y and Z” are then calculated from Boolean operations on the bit arrays. Some limitations have been found in this approach, notably the timeliness of convergence of

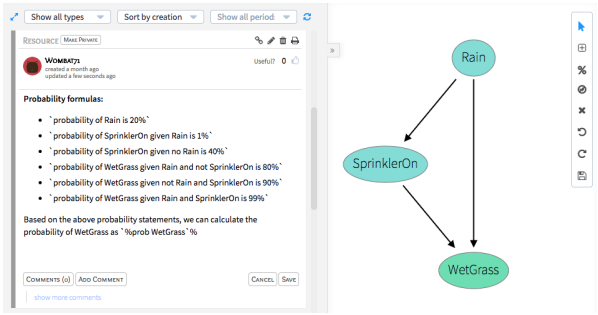

Figure 4. In-built Platform Features: Probability Calculator and Graphical Editor

results, especially on problems with a moderate number of variables, or on very small or large probabilities. Design is underway to address these limitations, while keeping the simple user interface. It is also noted that the LensKit also identifies a range of richer (and more complex) tools and environments for performing Bayes-type analytics off platform.

In addition to the probability calculator, SWARM recognized that various analytical approaches involve a visual aspect. Often a line of argumentation can be communicated more effectively when illustrated with a visual representation such as a dependency or causal loop diagram. Many critical thinking skills can also be improved through argument mapping. While users are free to use any graph tool they are familiar with and simply upload images to their responses, SWARM also developed an in-built graphical editor. This tool not only allows users to create any number of diagrams but it also lets users profit from a tight connection to the rest of the portal interface. The graphical editor can be opened from the response panel, which will turn the chat panel into a drawing canvas where diagrams can be constructed. Once finished the diagram will be saved right back to the current cursor position as part of the response. It is also possible to copy and paste text from other parts of the interface into the graphical editor, e.g. parts of the problem description or other responses. To support argument mapping, marked up text in the users own response can be imported and included into the graphics. Furthermore, the editor can also read in probabilistic expressions from the corresponding responses and automatically create a simple Bayesian diagram based on the calculations as shown in Figure 4 (right).

The graphical editor offers a range of basic diagramming features. The size, shape and colour of nodes can be changed and text in nodes and on arrows marked up. Nodes can be arranged on the canvas either manually or using a force-directed automatic

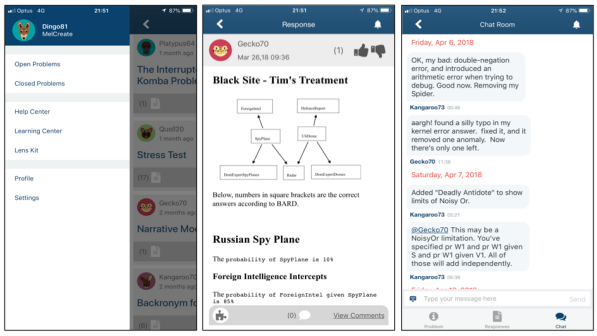

Figure 5. Mobile application

layout. Images can be dropped onto the canvas to become part of an existing node or to create a new node. Reverting changes and deleting functionality makes it easy to try out different approaches. The graphical editor was designed to be useful for an array of different use cases, e.g. from drawing a simple diagram to automatically creating a structure based on marked up text in the corresponding response. Further refinements and extensions to the graphical editor are under consideration, whilst acknowledging that many users will often want to use their graphical editor of choice outside of the SWARM platform and simply incorporate the images.

In addition to the portal based offerings, SWARM recognised that there are several fundamental challenges that need to be addressed for the SWARM platform to be successful in improving evidence-based reasoning. Arguably the most important of these is in avoiding attrition of users. There is no mechanism that can be applied to *make* users use the system. Rather, the system needs to be both easy to use and ideally encourage users to want to use the system. Nowadays, individuals will typically expect to access web content on their phone. This should allow ubiquitous access to most if not all of the features that exist within the SWARM portal. To tackle this, targeted SWARM mobile apps have been designed and delivered through the AppStore (iPhone/iPad) and Google Play (Android). Example of the features of the mobile apps are shown in Figure 5. The left pane shows the open/closed questions and the profile of the users. The middle pane shows the basic work panel related to a particular problem whilst the right panel shows the chat room. Unlike the portal which has all three panels available at all times, the mobile apps requires users to navigate across these panels (due to the obvious limitations of the screen space).

These apps enable users to access the SWARM content at any time. Users can receive notifications on the phone, e.g. by being mentioned in the chat room or when posts have been shared by team members. Reminders about ratings when a problem is nearing the end of its time window can be sent. This latter point is important since the quality of reports depends on the crowds engaging. Even if individuals in the team have not had chance to directly engage in the development of the responses, they can all read and rank the responses. This gentle encouragement is key to the adoption and success of SWARM. Whilst the apps can be downloaded from the AppStore/Google Play by anyone, only those that have a valid account on the SWARM platform are able to access and use the apps. This requires users enter the credentials that they have been assigned to access the platform.

6. Preliminary Results

The SWARM platform is currently in the process of being formally evaluated as part of the CREATE program by a Test & Evaluation (T&E) team with a crowd selected and managed by them, hence official results and feedback have not yet been produced. It is noted that the CREATE crowd for all platforms is expected to be over 4000 users with different numbers assigned to each platform and with different teams sizes. The SWARM platform has been deployed to the Amazon Cloud and integrated into a web-based front end (www.createbetterreasoning.com) together with all other CREATE-funded platforms for this official evaluation. Unofficial experiments have been conducted with SWARM for the last year however, using multiple releases of the platform on the NeCTAR Research Cloud. This involved multiple problems and multiple teams. The problems that have been set are representative of the kinds of problems that are to be set by the T&E. These problems have an associated set of rubrics that are used as the basis for deciding on the quality of the reports that are produced, and hence on the quality of the reasoning that has gone into those reports. The SWARM experimental team are responsible for the creation and subsequent assessment of the reports that are produced through the SWARM deployment on NeCTAR. The teams themselves have ranged from groups of 12-30 individuals with a variety of backgrounds. These individuals were recruited based on a SWARM-specific social media campaign (over 4500 users signed up to use the SWARM platform on Facebook with over 530 actually included into teams and using the platform).

While we cannot generalize over the evidence-based reasoning from the SWARM trials we can show that different group dynamics develop depending on the problem. Figure 6 shows the dynamic that developed within one group discussing a problem in the first

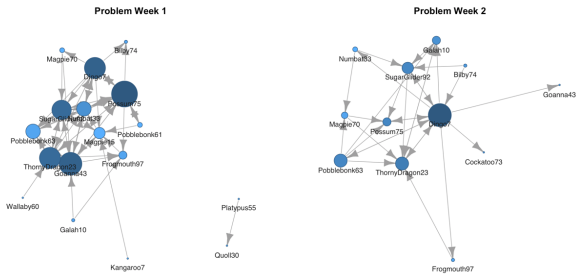

Figure 6. Visualizing Group Dynamics for a Given Team

week (left) and second week (right). For both graphs we collected the number of contacts between group members either by tagging each other in the chat or by commenting on each others responses. The size of the nodes indicates the social connectivity of team members (the number of contacts made), while the color shows their level of engagement on the system (the amount of responses, comments, chat messages and ratings they contributed). Both graphs show that there seems to be a correlation between both measures: large nodes are usually darker meaning that team members who contribute more also interact more with each other. As seen, a group of five people seem to be the most active players in solving the problem in week 1 whereas one core individual stands out in the second week. This difference might be explained by the fact that the problem posed in the second week required an understanding of Bayesian logic and the group relied heavily on a team member who seemed to have the necessary background.

7. Conclusions and Future Work

In this paper, we present the rationale and the design and development of the SWARM platform. The platform has been used extensively over the last year and the results appear very promising. The official assessment is currently ongoing by the CREATE T&E teams with the plans for adjudication of the platform to be made in the final quarter of 2018. Feedback on the platform by end users has been positive and most importantly, the reports that are produced are typically of high quality. Indeed the reports that are produced are often better than the official answers that have been prepared by the experimental teams.

Further work includes development of dashboards for individual/team analytics to better understand how and why teams interact to produce improved reports. Work on scaling the platform to deal with much larger teams is currently in focus as well as the challenges of web-based collaboration involving potentially hundreds of contributors. This will largely depend upon the successful evaluation of the SWARM platform.

Acknowledgments: This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research projects Activity (IARPA), under Contract [2017-16122000002]. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

[1] M. Li, "Internet of people," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 3, 2017.

[2] S. John Walker, "Big data: A revolution that will transform how we live, work, and think," 2014.

[3] A. Barnes, "Making intelligence analysis more intelligent: Using numeric probabilities," *Intelligence and National Security, 31(3)*, pp. 327–344, 2013.

[4] "A tradecraft primer: Structured analytic techniques for improving intelligence analysis," 2009.

[5] T. van Gelder and et al, "Cultivating expertise in informal reasoning," *Canadian Journal of Experimental Psychology, 58,*, pp. 142–152, 2004.

[6] M. Burgman, "Trusting judgements: How to get the best out of experts," 2016.

[7] A. Mannes and et al., "The wisdom of select crowds," *J Pers Soc Psychol, 107(2)*, pp. 276–299, 1997.

[8] D. Brabham, "Crowdsourcing," 2013.

[9] Y. Suzuki, "Assessing the quality of wikipedia editors through crowdsourcing," *Wiki Workshop 2016, Montreal, Canada*, 2016.

[10] L. Hong and S. Page, "Groups of diverse problem solvers can outperform groups of high-ability problem solvers," *Proceedings of the National Academy of Sciences of the USA*, pp. 16385–16389, 2004.

[11] D. Budescu and et al., "Identifying expertise to extract the wisdom of crowds," *Management Science, 61(2)*, pp. 267–280, 2015.

[12] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft, "Building reputation in stackoverflow: an empirical investigation," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, pp. 89–92, IEEE Press, 2013.

[13] Z. Kozhirbayev and R. O. Sinnott, "A performance comparison of container-based technologies for the cloud," *Future Generation Computer Systems*, vol. 68, pp. 175–182, 2017.

[14] D. K. Rensin, *Kubernetes - Scheduling the Future at Cloud Scale*. 2015.

[15] D. Hardt, *The OAuth 2.0 Authorization Framework*, 2012-10. https://tools.ietf.org/html/rfc6749.