Engineering Agent Organisations in a Business Environment

Dimitris Traskas and Julian Padget

Department of Computer Science
University of Bath, BATH BA2 7AY, UK
dtraskas@googlemail.com, jap@cs.bath.ac.uk

Abstract. Motivated by demands from the commercial world for software systems that can assist in the reorganisation of processes for the purpose of reducing business complexity, we discuss the benefits and challenges of the multi-agent approach. We concentrate on the engineering aspects of large scale multi-agent systems and begin our exploration by focusing on a real world example from the call centre industry. The critical call routing process seems appropriate and useful in presenting our ideas and provides a good starting point for the development of agent organisations capable of self-management and coordination. The main contributions of this work can be summarised as the demonstration of the value of agent organisational models that do not replicate the typical hierarchical structures observed in human organisations and that a quite basic peer-to-peer structure produces very similar performance indicators to a mature simulator that uses conventional techniques, suggesting further improvements may readily be realized.

1 Introduction

Over the last few decades the commercial world has become increasingly complex in an attempt to respond to rising consumer demands for more competitive products and services. Existing systems need to adapt and evolve rapidly in order to become more effective and flexible resulting in highly dynamic environments. Many businesses attempt to solve the complexity problem with simplification; essentially reducing the number of products and services offered or even the size of their customer base. The desired outcome of this approach is to simplify the processes and systems in place—however reduced performance and profits can also result.

In this paper we attempt to tackle the problems identified directly using multi-agent systems as the base technology and formal models of organisations as the informing theory in order to develop prototype solutions capable of operating in such dynamic environments. Our long term goal is the development of large scale agent societies interconnecting different areas of the business and capable of delivering their inherent benefits. The effective application of these systems in the business world depends heavily on three key factors: (i) the general framework that will be used to develop and deploy them, (ii) the modeling approach followed during the design process, and (iii) the organisational models used.

We explore the multi-agent approach using a real world example from the call centre domain and client data provided by our sponsor. Concentrating on the critical call routing process we provide a brief analysis of the problem and the key business objectives. Since agents are well suited to represent self-contained, autonomous modules of software capable of making independent decisions and taking actions proactively we investigate their use in developing the dynamic organisations required. An initial architecture is modeled based on the obvious hierarchical structure of call centres and compared by means of simulation with a decentralised alternative using the key performance indicators (KPIs) common in this domain. The hierarchical model is required to establish a baseline with which we can develop comparisons when working with alternative designs.

The remainder of this paper is laid out as follows: in the next section (2) we outline the business case for a multi-agent system in the call centre domain succeeded by (3) the modeling approach followed and the organisational models developed. In section (4) we present the experiments and metrics used to validate and present the performance of the agent systems developed. Section (5) presents comparisons of the various metrics derived from running the simulations on synthetic and real-world datasets and discusses our findings. Finally section (7) summarises the main conclusions of the paper and outlines future work.

2 Call Centre Case Study

Call centres are fundamental to the operation of numerous large organisations, emergency and government agencies and all types of customer service providers[9]. They constitute a vital component of and interdependency between many economies around the world, employing millions of people. As such, improvements in their management and performance would seem likely to have a significant impact. WorkForce Management (WFM) systems and a number of techniques borrowed from queueing theory through to artificial intelligence and simulation are used to forecast future inbound call volumes, allocate shifts efficiently or experiment with alternative business models. All along, the aim is to optimise performance and maintain the desired service level while preserving a balance between costs and service.

However call centre management and critical aspects of it such as the call routing process which we focus on in this paper, are becoming increasingly complicated due to the growing complexity of businesses. Call centres are getting bigger, physical colocation is becoming impractical and there are more interacting factors that complicate decision-making. Often the effects of poor call centre management are perceived as unsatisfactory levels of service offered to customers, inefficient allocation of resources which leads to increased running costs or high staff turnover.

We believe an appropriate response to these problems is to enable the construction of distributed virtual call centres, that take on much of the responsibility for their organisation and management and are able to adapt both to changes in load (environment) and changes in requirements. Control should become expressible through KPIs and QoS relationships and not by directly modifying the rules of the processes in place. Furthemore evaluation of alternative business models should become practical and cost

effective with the virtual call centre serving the purpose of an experimentation testbed capable of simulation and application. The overall characteristics of multi-agent systems motivated us to investigate their use as the base technology for implementing the software systems required. In addition the MAS paradigm seems appropriate for conceptualizing and designing highly dynamic models that can operate across multiple business and technology domains.

In the next few sections we analyse the call routing process and discuss its role in meeting business objectives. We subsequently demonstrate the agent-based modeling approach to the problem followed by the implementation of the prototype systems.

2.1 Call Routing Process

The term 'call-routing' is probably most commonly associated with telecommunications networks, where the task is to avoid hot-spot creation, maximise throughput and minimise latency. Popular approaches to this problem to include swarm routing [4] and stigmergic optimisation [5].

Call-routing in call centres is a somewhat different problem that is more akin to distributed resource discovery and allocation. Conventional implementations of call-routing in call centres are tightly controlled, centralised systems of asynchronous components, where all the decision-making is embedded in a single element—the call router—that communicates with the call-handlers (typically known as agents in the call-centre literature: here we use the term "handler" to alleviate confusion with the term (software) agent).

During the call routing process a call arrives from the telecommunications network and is ready for allocation by the 'Automatic Call Distributor' (ACD) part of the 'Computer Telephony Integration' (CTI) system. The ACD processes a set of business rules which determine which call has the highest priority. These rules use the call type or skill – as it is usually referred to in this industry – to identify the work with the highest criticality for the business in terms of performance. Another parameter used is handler availability, the resources waiting the longest without work will be at the top of the allocation list excluding those on breaks. The 'longest available handler' algorithm is common in most call centres although the simpler random allocation is not unusual either. If there is no available handler the call is inserted into a queue of calls that are ordered by waiting time and skill priority. Whenever a new resource becomes available the ACD gets notified and begins the allocation process for the list of calls currently in the queue. If customers wait too long and terminate their link with the ACD the call gets registered as abandoned and removed from the queue.

It becomes apparent that the ACD is a critical component of the call centre and any change requires expert knowledge and experience. Resource planning teams are able to manipulate the call routing scripts in order to improve performance and employ a number of techniques such as simulation and queueing theory (Erlang-C) to experiment with alternative operational models. However we have observed a number of problems through personal experience in this industry. Many call centres operate within tight time scales and budgets and are managed by poorly trained management staff. Collecting accurate data that will assist managers during the planning process is a difficult task and

depends heavily on reliable data sources and reporting tools. It is common to have data misinterpretations or encounter legacy software that cannot coexist with newly installed systems. As a result it is harder to develop precise simulation models and inform the decision making process. Additionally the centralised nature of the Call Router presents a single point of failure and a bottleneck with the increased demand of requests by the business.

We aim to address the problem from the MAS perspective and adopt agents in representing the key elements of the call routing process. Our objective is to implement a prototype system that can operate for application and simulation while using the same rules for resource discovery and allocation. Essentially business logic and rules in the form of java classes are injected within the agents and the mode of operation – application or simulation – does not affect that logic. Thus we expect improved data collection and accuracy while planning teams will have available an environment to experiment with the distribution of resources and skills or allocation rules.

3 Modelling Approach

We have encountered a number of challenges in engineering the prototype agent-based solutions we are presenting here primarily due to their scale. Borrowing techniques from the object-oriented methodology we identified the key elements of the call centre to develop a conceptual model whilst satisfying our goal of a scalable, autonomous system. The key characteristics of the call centre model are summarised below:

- The *Call Centre* consisting of customer calls, handlers and routing information.
- The *Call Distributor* or Router which receives calls and routes them to the human agents.
- The *Call* which is a packet of information used by the Call Distributor and which contains customer details, type of call and time of arrival.
- The *Skill* which is the type of call that a call handler can handle, e.g technical support.
- The *Skill Group* a grouping mechanism used to define the different skills call handlers may have e.g an insurance skill group consists of motor, home and pet insurance.
- And finally the *Call Handler* a trained worker with a set of skills or skill group able to answer customer requests.

Using this model we then designed and implemented an initial prototype multi-agent system. It is clear that the fundamental parts of the call routing process are the handler and router. These two key elements can be represented by an agent and interact as they would in the real world. An agent operating in this environment communicates using a simple set of messages such as ‘accept call offer’, ‘reject call offer’ or ‘send confirmation’. Within those messages we include useful information about the caller, the time or type of call and propagate it using the message transport mechanism.

Early on we identified a number of problems with the hierarchical approach mainly due to the large number of messages being sent. For all agents to perceive the current

state of their environment it was necessary to propagate information to each and every one of them. This introduces two problems, the design depends heavily on the router agent which now becomes a bottleneck and a single point of failure and also the number of messages required is now increased dramatically.

A decentralised architecture however could potentially consist entirely of handlers and not require the router agent. Handlers in this design are servers and can route the calls to the appropriate agent colleague if they behave in an unselfish manner. Nonetheless with this design there are important questions to be answered:

- Where can we store and maintain the queue of calls?
- How do we ensure that the last available handler receives the highest priority call?
- What happens when a call gets abandoned? Can we inform all the handlers of that event?
- What if two handlers decide to handle a call at the same time? Which is the entity that will prioritise?

The research area of peer to peer networks [1] provided us with a number of ideas although not directly applicable to our problem. The concept of a peer is almost indistinguishable from that of an agent so any design patterns used in this area could be useful to us too. An interesting concept is that of peer rings where information propagates in a cyclical manner. Potentially call information could be propagated this way and system integrity ensured. Another area we had some inspiration from was the work on semantic overlay networks [3] and the idea of cluster formations based on certain agent properties. A handler agent that can only answer calls of a specific type seems like a reasonable property to use for that purpose. Using the skill information we tag agents and during an initial bootstrapping phase allow them to self-organise in skill-based clusters.

In the next two sections we present both architectures in detail and present their key concepts. For the purpose of this paper we largely omit implementation details as we only need to expose a high level view of the systems developed.

3.1 The Hierarchical Model

The initial and apparently obvious architecture for the call routing process is essentially a consequence of earlier human organisational structures: a telephone operator receives calls and then routes them to the longest available handler. However, the pattern it implements is intrinsically a function of the strengths and weaknesses of *human* agents. Nevertheless, an architecture that does not conform to the expectations of stakeholders may face difficulties in being accepted.

The hierarchical model mimics the current allocation process and requires three core agents: **Administrator**, **Router** and **Handler**. The Intelligent Call Distributor (ICD) as it was initially termed, is potentially capable of operating for application and simulation. The difference lies heavily on the deployment process and the capabilities of the agent platform used.

Exploring the routing process even further we identify a Router agent receiving customer calls in random intervals and using the routing information defined in the

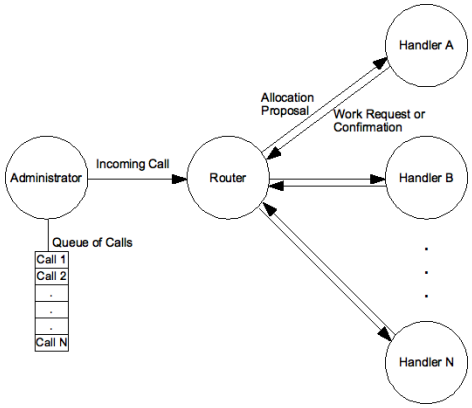

Fig. 1: Overview of the ICD architecture.

model to search for an available handler. This routing information is in the form of skills, skill groups and priorities as defined by the business. Once an appropriate handler agent is found a negotiation begins with a proposal. This negotiation either leads to an allocation or the continuation of the resource discovery process. The Router queues calls that it cannot allocate to a handler and waits for the next cycle of processing. In a real world scenario a customer waiting too long will abandon the call. We account for this behaviour using a patience time model which we have developed based on the exponential distribution [10] appropriate for this kind of simulation delays. For each call that arrives we generate randomised abandonment times using the formula below:

$$\text{Time in seconds} = (-\ln(1-r)\sigma)$$

where r is a random number between 0 and 1 and σ is the average patience time. The same model is considered suitable to generate the handling times of agent handlers. Eventually the Router will check for calls that have reached their patience time and abandon them. Our design requires the Router and Handler agents to be synchronised once an event such as a customer call arrives, achieved with the use of Finite State Machines (FSM). Below we describe in summary the key states of the Router and Handler agents.

Router Agent

1. *Receive model:* The Administrator sends skill / skill group definitions and the total number of handlers in a message.
2. *Receive call:* The Router waits for an incoming call from the Administrator.
3. *Receive work requests:* Handler agents send work requests to the Router with attached current availability, working hours and the set of skills available. The Router requires work requests from the total number of Handlers to be received in order to proceed to the call allocation phase.

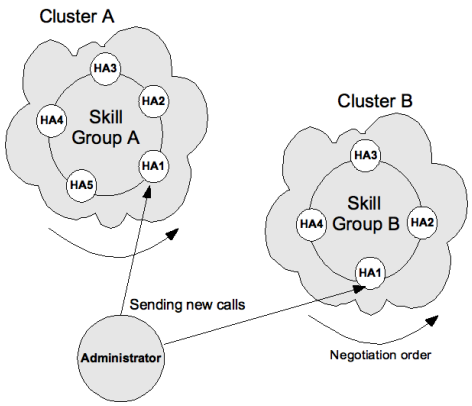

Fig. 2: An example IRN with two skill groups and clusters.

4. *Allocate call:* Once all work requests are received, the Router orders the list of Handlers by availabiltiy and attempts to allocate the top call in the queue by sending a proposal message to the Handler.
5. *Receive confirmation:* If a Handler is able to handle the work proposed and available – not on a break – will send back a confirmation message otherwise a rejection. In the latter case the Router will attempt to allocate the work to a different agent following the procedures of state (4). Calls that cannot be handled are added in a waiting queue.
6. *Check abandonment:* In this final state the list of abandoned calls is checked and any call where patience time has been reached becomes abandoned.

Handler Agent

1. *Initialise:* Similar to the initial state of the Router the agent requests skill and shift information.
2. *Request work:* Handlers send work requests to the Router even if there is a possibility that they will still be busy when the next call arrives.
3. *Confirm or Reject:* If a proposal message is received, the Handler will evaluate the work offered and reply either with a confirmation or a rejection message.

3.2 A Decentralised Approach

In the absence of a Router, the Intelligent Routing Network (IRN) requires Handler agents to coordinate and find the longest available handler with the right skill to handle a call. IRN uses a semantic overlay network of skill group based clusters to allocate calls. Clusters and connections are formed during an initial bootstrapping sequence using the tagging mechanism first proposed by Holland [8]. Essentially handlers are tagged based on their skill information and grouped in clusters.

The system uses two core agents: **Administrator** and **Handler**. The key characteristics of this architecture are: (i) the formation of clusters able to handle different skills, (ii) call queues per cluster, (iii) a negotiation cycle required to process every event, (iv) a forward list used to transfer information from one agent to another, (v) state replication across all agents in a cluster.

Handlers are linked in a circular list where one agent is assigned the role of the *first* agent in the cycle. This agent is required to receive a new call from the Administrator and complete a cycle for every action. When a call arrives the *first* Handler will ensure to update the local queue of calls and thus maintain the same state across all agents in the cluster. This is to prevent loss of useful routing information when an agent or host fail and become disconnected from IRN. The forward list is a simple mechanism which allows agents to wrap any information required for state updates, call allocation and call abandonment. On completion of the update process the agent will check availability and immediately send the forward list and current call to the next Handler. Once the message arrives back to the first Handler a cycle is complete and the information contained in the forward list provides the longest available agent and call to be allocated. This cyclic processing of events is required to gather all information necessary to allocate, queue or abandon a call and ensure that queues of calls are replicated across all agents in the cluster. For the purpose of page economy we present only the key states of the Handler agent below.

Handler Agent

1. *Initialise:* The agent requests initialisation data.
2. *Ping:* The agent starts sending ping messages to all agents in the network.
3. *Pong:* On receipt of a ping message the agent replies with pong and its skill group attached.
4. *Cluster formation:* Once all agents in the network have replied the agent uses skill group information to form a cluster.
5. *Process new call:* The first Handler in the negotiation cycle receives a new call and checks the skill and current availability. The local waiting queue gets updated accordingly and ordered by the longest waiting call. All this information is forwarded to the next agent in the cycle.
6. *Allocate call:* When a call completes one negotiation cycle information collected is used to determine who is the longest available Handler. A new negotiation cycle begins to clear all local queues from the call and allocate it. If there are more calls to handle and available agents a new cycle begins to allocate it, otherwise calls are tested for abandonment. A negotiation cycle is also required to abandon-clear calls from all queues.
7. *Process forwarded call:* On receipt of a forwarded call Handler availability is tested and added to the forward list. If this is the last agent in the cycle then the information collected is used to allocate the call.
8. *Abandon call:* When a customer's patience time is reached a call becomes abandoned and added in the forward list which is forwarded to the cluster.

4 Experiments

For validation purposes we conducted a number of simulation experiments with different scales and using synthetic and empirical data. We compared the performance of the two solutions with that of an industry-standard call centre simulator from CACI Ltd[1] called Call Centre Workshop (CCW) and actual data provided by one of CACI's clients. CCW is a commercial product used by a significant number of clients from the software, retail, banking, insurance and mobile phone sectors. For our experiments we used a set of performance metrics to validate and compare the results such as Service Level (SL%), Calls Handled and Calls Abandoned, Queue Length and handler Utilisation per interval [9]. These are the standard metrics used by call centre managers and planning teams to measure business performance and assist in the decision making process. Service Level is the percentage of calls answered within a business-specific time frame. This time frame is called Telephone Service Factor (TSF) and is usually in the range of 20-30 seconds. Service Level can be used to measure intra-day, daily and weekly performance and is normally defined as:

$$\text{SL\%} = \frac{\text{Calls Answered before TSF}}{\text{Calls Answered} + \text{Calls Abandoned}} \times 100$$

Utilisation is defined as the percentage of time handlers that are busy per interval in a day and Queue Length defined as the number of calls queued per interval. Other metrics that are common in the call centre domain are Average Handle Time (AHT), Average Patience Time (APT) which is the average time customers wait on the phone before they abandon and Average Answer Time (ASA). We created synthetic datasets with an aim to test the agent prototypes and allow all different scenarios to be handled by the agents. An example synthetic model used has four skills, variable call density and shift breaks during afternoon hours in order to demonstrate the expected drop in service level and the increase of abandonment. The attributes of the model used can be seen in table 1.

Table 1: Synthetic Model (4 skills, 10 handlers, 1420 calls)

Skills	TSF (secs)	Priority	AHT (secs)	APT (secs)
LOANS	20	1	120	180
MORTGAGES	20	1	180	240
PET INSURANCE	20	1	120	360
MOTOR INSURANCE	20	1	240	120

The second phase of the experimentation process was the use of real customer data. We have data available from a number of clients but in many cases the computational cost of running such simulations is prohibitive. Typical scenarios require 5,000 handler agents and 400,000 calls in a day just to simulate one run of the call allocation process.

[1] `http://www.caci.co.uk`

The call centre selected for our experiments is significantly smaller and part of one of the UK's leading mobile phone retailers. The client provided us with actual data from one of their main call centres which we use to simulate one day with 560 Handlers, 43,365 incoming calls and 13 different skills. Calls received are considered to be within service level if they have been answered within 20 seconds from the time of arrival. Skill handling times were varied through the day, while customer patience times are estimated on average to be 180 seconds.

For the purpose of this work we set-up a very simple call routing model with no skill priorities and a direct mapping of skills to skill groups. We then loaded the data mentioned above in ICD and IRN and repeated the exact same process in CCW. The average handling and patience times were used to generate randomly the time it takes for a handler to handle a call and a customer to abandon it using the exponential distribution models discussed earlier. We executed the simulation for 5 runs and compared our results with those of CCW as well as the actual service level values. The client's resource planning team calculated the actual service level values after collecting all the data necessary stored for that day in the ACD database and using the same routing model with us. Unfortunately they were not able to provide other performance metrics such as utilisation or queuelength that would be very useful for our experimentation thus we only demonstrate a comparison of service level.

4.1 Call Centre Workshop (CCW)

CCW allows the user to develop simulation models of call centres and specify through its user interface skills and skill groups, the number of handlers working on every period, average handling and patience times and call allocation rules. Using a standard workflow language CACI has developed a library of workflows for call centre processes that are used in the simulation depending on the routing and queueing selected. When the simulation begins: (i) handler resources are created, (ii) random calls generated for each interval based on the specified call density, (iii) a routing workflow selected and (iv) a simulation loop initialised. Each new call from the list is treated as a discrete event and sent to the workflow for further processing. Through that process the call is checked for skill, skill priority and sent to the longest available handler for allocation. Any call that has not been allocated is queued until the next event comes and the clock moves forward. An important step in the workflow is the check for abandonment, essentially if the customer has reached her patience limits the call becomes abandoned. During the simulation a Data Collector retrieves low level raw data that is subsequently parsed to produce the performance metrics described in earlier sections. The routing workflows used are standard for most simulations however CCW allows modelers to customise the process based on business requirements. The ICD, IRN and CCW simulators follow exactly the same business rules and constraints required for our experiments in order to prove that the agent approach is equally effective.

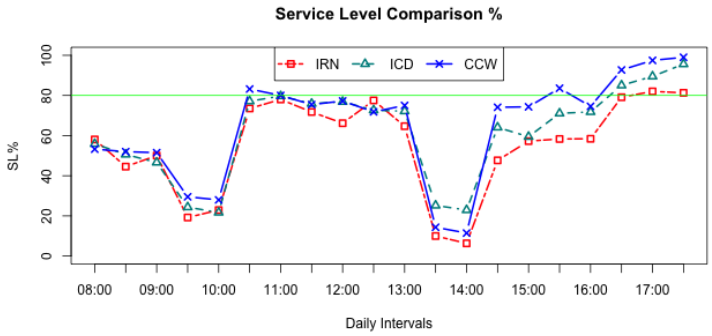

Fig. 3: Service Level plot of hierarchical and decentralised agent-based systems compared with CCW.

5 Results and Discussion

Below we present comparisons with the metrics produced from the synthetic model experiments. The graph in figure 3 shows changes in service level throughout the day with a target service level of 80% represented by the green (horizontal) line, a common target across many businesses and sectors. We wanted to demonstrate that the systems behave as expected when incoming call volumes remain at high levels at certain periods in the day while at the same time agents start their breaks. As it was anticipated we notice a dramatic decrease of service during early morning and afternoon hours. The overall variance of the daily average of service level between the ICD and CCW was 3% and between IRN and CCW 9.6% which we believe is within the acceptable range of 3%-10% of an operational model[11].

To further validate our model and confirm that the systems behave as they should we also calculated the number of calls handled and abandoned through the day as we can see in figure 4. We notice that between 13:00 and 14:30 the number of calls handled drops significantly, that is because we have reduced the number of available agents by 60% at a very busy time in our hypothetical call centre. However the number of calls abandoned increases during that same period reflecting the inability of the business to service customers queueing for a very long time. The Resource utilisation plot demonstrates that agents are generally very busy due to high volume of inbound calls and, as expected, their availability drops between 13:00 and 14:30.

Finally the results shown in figure 5 demonstrate the performance of ICD and IRN using the real world model and, we believe, they show a satisfying variance of 7.5% between the daily average service level of the ICD and actuals and 5.8% between the IRN and actuals. The prototypes appear to have sufficient accuracy considering the complexity of the problem, the volume of data and the unknown parameters of the client's routing process. It would be very useful of course to use more metrics as service level is a high level indicator and can sometimes hide some of the detail required.

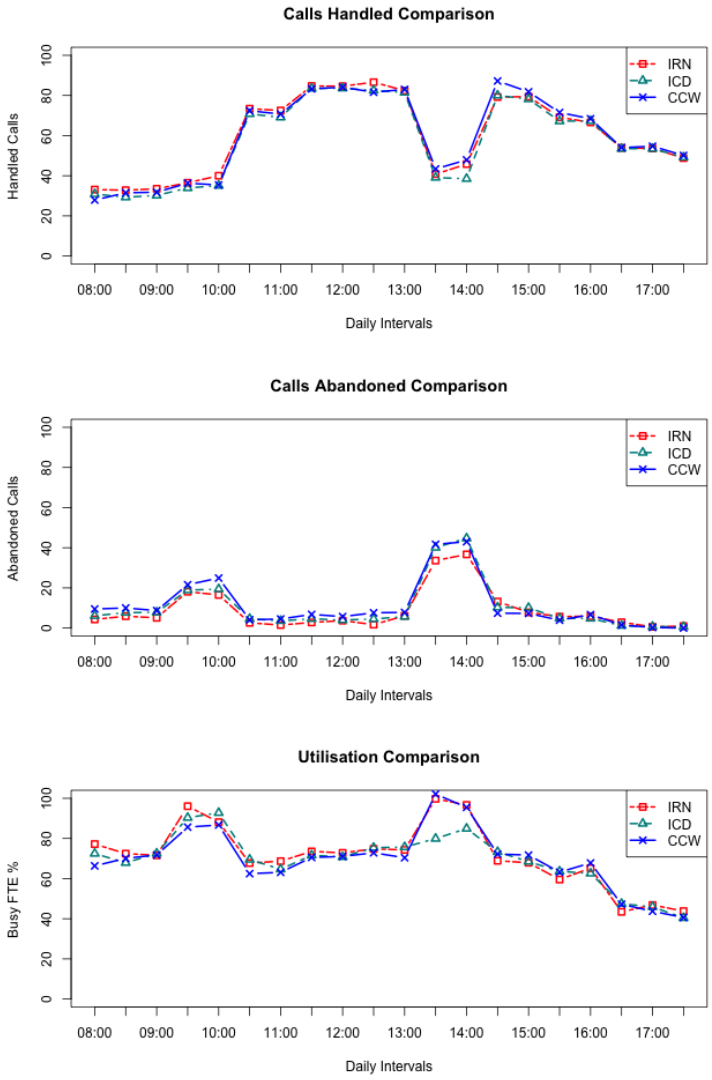

Fig. 4: Calls Handled, Abandoned and Utilisation comparison plots.

The results presented in this section confirm that the multi-agent prototypes developed are effective and inspire us to further experimentation. There are however inefficiencies that became apparent during the course of this study. The frameworks available for the development of large scale multi-agent systems are limited. We begun our development with an object-oriented design of the call centre which allowed us to identify the key agents. It would be useful during that phase to model the interactions of the

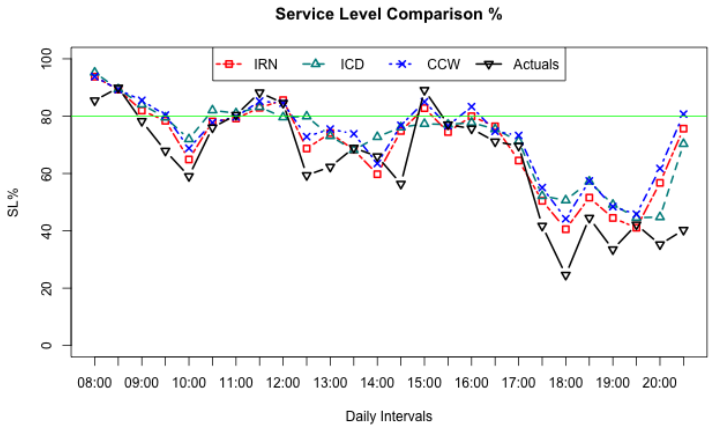

Fig. 5: Service Level plot of hierarchical, decentralised, CCW and actuals.

agents outside the agent platform and detect problems with the logic and state of the agents that could bring the system to a deadlock or race condition. We lack the modeling tools and methods that will allow not only us but also high level users of such systems to manipulate and investigate the agents without coming into contact with the low level implementation details.

Our first hierarchical prototype was developed using the JADE [2] agent platform but due to scalability problems we continued our work with Cougaar which stands for 'Cognitive Agent Architecture' [7]. Cougaar is the product of an eight-year DARPA-funded research project and developed to meet demanding scalability, reliability and survivability needs. Agents in Cougaar contain plug-in components — rather than behaviours — and interact with each other via a blackboard-based publish and subscribe mechanism — rather than messaging. The blackboard encourages the use of events for agent interaction and is the primary mechanism for state management and message exchange. The plug-in software components provide a domain-specific behaviour to the agent and can be added or removed at design or run-time. A simple relay mechanism allows the communication across agents spawned within different host machines and through the blackboard. For the purpose of our work we created different types of messages and packets of information that could be wrapped in a single message. Each agent has to subscribe to the different message types and listen for incoming messages. The platform so far has proven to be highly scalable and allowed us to deploy the real world call centre model with 560 Agent Handlers. Its threading pool model and resource handling is more rigorous than other platforms that we have examined and encourages future use. Below we provide a summary of our findings:

– We have shown that a multi-agent architecture can be used effectively to manage a business process with a real world example from the call centre industry. The agent systems can allocate inbound call volumes to resources while adhering to

business rules and constraints. An inherent advantage of the proposed solutions is their dual purpose. They can operate for application and simulation offering to businesses an experimentation framework that will assist in the decision making process by evaluating alternative operational models without the imposed risk and cost of current approaches.
- We have demonstrated an approach where system architecture does not need to follow human intuition which encourages the use of a hierarchical model. Instead the solution should take advantage of the properties of agent-based organisations of self-coordination and self-management.
- The inefficiencies that become apparent with the scale of the experiments conducted indicate the need for further investigation of organisational models and frameworks. The current state of MAS technology prohibits the deployment of such systems within a commercial environment with the aim of upgrading existing infrastructure.

In general we believe that the multi-agent paradigm has many benefits to offer to the corporate world but it is imperative to look into the problems and challenges of the technology and devise new methodologies and organisational models. Although the domain of MAS advocates scalability as one of its benefits it is very difficult to find the frameworks, supporting methodologies and tools required to develop these large scale systems.

6 Related Work

Multi-Agent Systems have long been identified as a suitable framework for the development of the virtual enterprise [14]. The effective application of agents in the commercial world however has been minimal and even today with the advance of internet technologies and increase in computational power we have few examples to demonstrate. We haven't found many examples of MAS in the call centre sector and much of the research work discusses agents as a useful abstraction for modeling business processes.

The work described in [13] concentrates on the ADEPT multi-agent architecture which allows the coordination of agents across different business domains using negotiation. ADEPT operates on the basis of transforming existing business processes into a number of agencies under a hierarchy. The hierarchical structuring of agencies enables the encapsulation and abstraction of services. The example used does not need to identify the key performance indicators that are critical in the business process simulated compared to ours and does not discuss how agent societies would organise in large scale corporate environments.

In [12] the authors discuss the use of a meta-layer to assist in system adaptation without requiring any global information. A semantic overlay network is created on top of the lower level agents, however the specific tasks and roles agents can have are not necessary to establish their organisation. This is a useful approach for some problems but in the demanding business environment it is crucial to specify the exact business rules and tasks of resources such as the Call Handler. Any system deployed in a corporate scale would have to deliver good measurable performance whilst being capable of satisfying business rules and constraints and adapting to ever changing demands.

Finally in [6] we are presented with an overlay system of controller agents associated with application agents. Controllers can detect rule violations based on their observations and sanction the violators by using reputation. The main difference with the call centre example is that in our designs agents are tightly controlled and have limited freedom in their decision making process, as it is required by the business. We plan however to experiment with alternative organisational models in the future where agents have a lot more freedom to interact with each other whilst delivering good overall system performance.

7 Conclusions and future work

This paper advocates the multi-agent approach in the corporate environment in order to create software systems that are capable of self-organisation and management, adaptable to the ever changing requirements of businesses. More specifically we support the case for an alternative model in call centre management and investigate the call routing process, a critical operation in this domain. We develop two prototypes, a hierarchical system as a baseline model that mimics the current routing process and a decentralised one that consists of self-organising and coordinating handler agents. Using synthetic and empirical data we conduct a number of simulation experiments and show that the hierarchical prototype is capable of performing equally well with the conventional call router. The centralised manner of this design however exhibits a number of problems such as inability to scale better, a single point of failure – the Router – and incapable of adapting to changes in the process. Influenced by ideas from peer to peer computing and semantic overlay networks we progress our work and implement a decentralised routing network capable of handling call and handler prioritisation and queueing calls without the need for a central control entity. Following the same experimentation methodology we prove that this design is also equally effective when compared to our initial attempt although it presents a number of challenges.

Future work includes the investigation of alternative decentralised architectures that are flexible and efficient in tackling the problems mentioned in this study. We wish to relax the constraints and rules of our designs and experiment with agents that respond to changes in their environment using probabilistic models rather than the rule-based approach we currently employ. We also plan to conduct further research work on the methodologies and tools that will allow us to decrease development and debugging time. It is necessary to have a modeling framework that will help us produce and test new designs. Furthermore such a framework could be extended to become a high level tool for business managers who do not require to be aware of all the implementation details but wish to manipulate the logic and interactions of agents.

8 Acknowledgements

We wish to thank CACI Ltd for their support and for providing access to client data and the Call Centre Workshop software. We would also like to note that our findings and conclusions do not necessarily reflect those of the sponsor.

References

1. S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. In *ACM Computing Surveys (CSUR)*, volume 36, pages 335–371. ACM Press, 2004. ISSN:0360-0300.
2. F. Bellifemine, A. Poggi, and G. Rimassa. Jade: a fipa2000 compliant agent development environment. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 216–217. ACM, 2001.
3. A. Crespo and H. G. Molina. Semantic overlay networks for p2p systems. In *Agents and Peer-to-Peer Computing*, pages 1–13. Springer Berlin / Heidelberg, 2002. ISSN: 0302-9743. ISBN: 978-3-540-29755-0.
4. G. Di Caro, F. Ducatelle, and L. Gambardella. Swarm intelligence for routing in mobile ad hoc networks. In *Proceedings IEEE Swarm Intelligence Symposium 2005*, pages 76–83. IEEE Computer Society, 2005.
5. M. Dorigo, G. D. Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
6. A. Grizard, L. Vercouter, T. Stratulat, and G. Muller. A peer-to-peer normative system to achieve social order. *Coordination, Organizations, Institutions, and Norms in Agent Systems II: AAMAS 2006 and ECAI 2006 International Workshops, COIN 2006 Hakodate, Japan, May 9, 2006 Riva del Garda, Italy, August 28, 2006. Revised Selected Papers*, pages 274–289, 2007.
7. A. Helsinger, M. Thome, and T. Wright. Cougaar: a scalable, distributed multi-agent architecture. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1910–1917. IEEE, 2004. ISSN: 1062-922X. ISBN: 0-7803-8566-7. INSPEC Accession Number: 8393468. Digital Object Identifier: 10.1109/ICSMC.2004.1399959.
8. J. H. Holland. *Hidden Order: How Adaptation Builds Complexity*. The Perseus Books Group, 1995. ISBN-13: 9780201407938.
9. G. Koole. Call center mathematics: A scientific method for understanding and improving contact centers. Available from `http://www.math.vu.nl/~koole/ccmath/book.pdf`. Verified 20080520.
10. D. J. Lilja. *Measuring computer performance: a practitioner's guide*. Cambridge University Press, New York, NY, USA, 2000.
11. C. M. M. Michael J. North. *MANAGING BUSINESS COMPLEXITY — Discovering Strategic Solutions with Agent-Based Modelling and Simulation*. Oxford University Press, 198 Madison Avenue, New York, New York 10016, USA, 2007.
12. J. C. Miralles, M. López-Sánchez, and M. Esteva. Multi-agent system adaptation in a peer-to-peer scenario. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 735–739, New York, NY, USA, 2009. ACM.
13. T. Norman, N. R. Jennings, P. Faratin, and E. H. Mamdani. Designing and implementing a multi-agent architecture for business process management. In *In Intelligent Agents III*, pages 261–275. Springer-Verlag, 1997.
14. C. Petrie, C. Bussler, and A. Survey. Service agents and virtual enterprises: A survey. *Survey, Internet Computing, July/August 2003*, 7, 2003.