Social Machines for All
Blue Sky Ideas Track

Petros Papapanagiotou  
University of Edinburgh  
Edinburgh, United Kingdom  
ppapapan@inf.ed.ac.uk

Alan Davoust  
University of Edinburgh  
Edinburgh, United Kingdom  
adavoust@inf.ed.ac.uk

Dave Murray-Rust  
University of Edinburgh  
Edinburgh, United Kingdom  
d.murray-rust@ed.ac.uk

Areti Manataki  
University of Edinburgh  
Edinburgh, United Kingdom  
A.Manataki@ed.ac.uk

Max Van Kleek  
University of Oxford  
Oxford, United Kingdom  
max.van.kleek@cs.ox.ac.uk

Nigel Shadbolt  
University of Oxford  
Oxford, United Kingdom  
nigel.shadbolt@cs.ox.ac.uk

Dave Robertson  
University of Edinburgh  
Edinburgh, United Kingdom  
dr@inf.ed.ac.uk

ABSTRACT
In today’s interconnected world, people interact to a unprecedented degree through the use of digital platforms and services, forming complex ‘social machines’. These are now homes to autonomous agents as well as people, providing an open space where human and computational intelligence can mingle—a new frontier for distributed agent systems. However, participants typically have limited autonomy to define and shape the machines they are part of.

In this paper, we envision a future where individuals are able to develop their own Social Machines, enabling them to interact in a trustworthy, decentralized way. To make this possible, development methods and tools must see their barriers-to-entry dramatically lowered. People should be able to specify the agent roles and interaction patterns in an intuitive, visual way, analyse and test their designs and deploy them as easy to use systems.

We argue that this is a challenging but realistic goal, which should be tackled by navigating the trade-off between the accessibility of the design methods—primarily the modelling formalisms—and their expressive power. We support our arguments by drawing ideas from different research areas including electronic institutions, agent-based simulation, process modelling, formal verification, and model-driven engineering.

KEYWORDS
Social Machines; Design; Analysis; Modelling; Model-Driven Development

1 INTRODUCTION
As an increasing share of human interactions have become mediated by software systems, a new class of socio-technical systems has emerged, sometimes called Social Machines [45, 46] or Human-Machine networks [14]. Examples include Wikipedia, Twitter, dating applications, citizen science platforms, Q&A sites, MOOCs, and many more. Autonomous agents are slowly acquiring meaningful roles in this space—one can think of the proliferation of bots on Wikipedia, Twitter, or Facebook, and many platforms are specifically aiming to support agents via rich APIs as part of their design (e.g. the Slack messaging platform or the Github code repository).

In a nutshell, we are witnessing the rise of open multi-agent systems on the Web, including both human and software agents, communicating via Web technologies.

The original conceptualisation of social machines was that machines would do the administration while humans were free to carry out creative goals, with an intention that this should give rise to “new forms of social process”[5, pp. 172–175]. This drive towards new forms of social process can be seen around the edges of platforms: when hashtags on Twitter emerge as coordination artefacts for sharing scientific knowledge (as in the case of #icanhazpdf1), or when blog comments becomes a vehicle for carrying out research activities (as in the case of /r/scholar subreddit) and on Facebook [19].

In this paper, we outline a research agenda to democratize the development of Social Machines, with the ultimate aim of supporting a wide range of people in creating mixed human-machine communities, where computational agents support human sociality.

1 Academics (and others) use this Twitter hashtag to request scientific articles which are inaccessible to them due to paywalls, and others share the articles by email or by responding with a link to an accessible copy. Similar activities happen on Reddit (/r/scholar subreddit) and on Facebook [19].

in an open, extensible manner. We view this initiative as a parallel to the push towards the re-decentralisation of the Web, in line with projects such as IPFS [4] and SOLID [11].

Social Machines are a frontier for open agent systems on the web, and as such we want a future where they are available to all.

Our Vision: Social Machines for All
We have been inspired by working with a range of people over several years to understand how they might go about developing social machines. The Cybermadres diagram (Figure 1) is a typical output from a design workshop. This informal sketch of a social machine was the result of an open-ended process, seeded with an initial set of diagrams illustrating interactions in a particular, simplified way, to be extended and subverted as necessary.

Despite its informality, the diagram conveys many relevant aspects of this social machine: we can see actors with different roles, coordination infrastructure, and sketches of interaction between them. Arguably, people familiar with existing social machines can easily interpret the diagram and visualize the finished system.

This demonstrates the ability of laymen to design social machines from a standing start in just a few hours. The next step to produce a working social machine is to create Web-based infrastructure to support it. Enabling the automatic transformation of such designs into working systems would massively enhance the possibilities for bottom-up, democratic creation of social machines. So what are the challenges to make such automatic transformations possible?

In order to do this, we need conceptualisations that are both formal and useful to novice designers, ideally in the form of a visual language that allows non-experts to describe social machines both at a technical and a social level. We then need automation machinery that can transform such designs into usable software, ready to be populated with human and software agents. We also need methods and tools for analysing, debugging and understanding social machines— from “unit tests” to ensure that the interaction patterns make sense to advanced simulation and verification techniques to validate technical and social properties.

To support our contention that this vision is not entirely out of reach, we identify three corresponding main research challenges, discussing the main promising avenues to address them.

2 GRAND CHALLENGE: FROM INTUITIVE MODELS TO DEPLOYED SOCIAL MACHINES
The central challenge in democratizing the development of social machines is bridging the gap between intuitive, comprehensible models and fundamentally complex systems. We envision the designers specifying only high-level patterns of interaction (e.g. “make announcement”, in the CyberMadres example) or coordination infrastructure (e.g. “a voting system”), and obtaining “out of the box” computational support for those elements, based on a limited set of building blocks.

Parallels exist with Model-Driven Engineering [18], which aims to make models, not code, the primary artefact of the software engineering process. High-level models are transformed into executable code, through the use of patterns, frameworks and templates. While this technique has been applied to multi-agent systems [1, 39], these approaches are intended for high expressiveness and result in extremely complex agent models. Similarly, MDD has been applied to Virtual Organizations [2, 3] and the systematic transformation of simple workflow models into executable code (e.g. UML activity diagrams into BPEL[20]).

By viewing hybrid system development as model creation, we can begin to decompose our overarching challenge into different sub-problems, organized around the modelling artefact.

(1) How can a layperson create intuitive models of complex agent systems? This requires simple representations with enough flexibility to incorporate social elements, such as motivations, incentives, influences, yet specific enough to provide a formal enough system that can be analyzed, simulated, and deployed.

(2) How should the agent coordination patterns produced from the model be deployed and monitored on the Web, particularly when both human and software agents are involved?

(3) What tools can be provided to test, debug and predict the success of social machines, creating a design cycle that includes social and technical assurance and feedback?

Our remaining sections unpack these into a series of challenges for different aspects of potential approaches.

3 CHALLENGE #1: CREATING INTUITIVE MODELS OF COMPLEX INTERACTIONS
We first consider the following challenge:

What modelling formalism could enable a non-expert to design a meaningful class of social machines?
MAS researchers are used to formalisms that describe agent interaction in a way that supports execution—for example Electronic Institutions (Islander [15], EIDE [34]), and organizational models based on norms (MOISE [22], OPERA [12]). In particular, recent efforts for hierarchical institution governance [24] attempt to specify the permitted norms of participation in a social machine in a way that fits well with the framework described here. Similarly Process Management researchers work with workflow specification languages (e.g. BPMN [36], BPEL [35]). While we are not arguing against the need for rich and complex languages such as these, for this challenge we are interested in extending existing practice in space of simple specifications, with the understanding that this will necessarily constrain the range of systems that can be described. Essentially, most serious development is at the complex end of the spectrum, covering intricate formal systems in unambiguous detail.

In contrast, we hope that formalisms exist where a small number of high-level constructs can be assembled like building blocks to describe complex interaction patterns, such that the proverbial 80% of applications is catered for with 20% of the complexity. Varying specifications of primitives exist: HTTP has a few verbs (GET, POST, PUT, DELETE etc. [16]) while Twitter is highly constrained, with just 3: TWEET, LIKE, RETWEET. In the agent engineering world, FIPA’s communicative acts specification has around 20 performatives [17] with complex compositional semantics. At the simple end of the scale, data science became massively parallelised with just two verbs: map and reduce.

It is therefore natural to pose the question: can we extrapolate from this observation and determine a set of primitives needed to build social machines? Central to this challenge is finding representative levels that are broadly applicable enough to be re-usable, yet precise enough to support execution. Saying an actor MANAGES a collection brings a host of intuitive semantics around create, read, update and delete operations; an N-way agreement between agents or RESERVING physical resources are concisely understandable, and re-usable components of complex systems. The appropriateness of different components may also depend on the particular type of social machine that is being designed. The selected building blocks need to accommodate different categories of social machines, for example based on known taxonomies [45, 46].

These primitives then need to be composed into larger scale machines—In Figure 1, actors need to ASK_ALL of the restaurants for excess food and then RESERVE it for collection. This requires a formal composition language that systematically links such processes by establishing, for example, information or resource flows and shared data structures on.

There is an array of theories and tools for supporting such composition. BPEL [35], for example, enables business process composition linking to web services and via executable semantics, but does not have a visual representation and supports centralized orchestration (as opposed to peer-to-peer choreography). Process algebras such as the π-calculus [29] and session types [23] offer solid theoretical foundations for the systematic analysis of distributed systems, but are far from intuitive for non-expert users. Recent efforts in formally verified process specification and composition [38] have taken steps towards visual process composition with formally verified properties and automatic code generation.

An additional challenge arises in maintaining the social side of these systems: motivations for participation, incentives [43], social norms and expectations [8], knowledge sharing etc.

This vision raises several key questions: first of all, what should a primitive look like and how to develop a usable set of primitives? Secondly, what does it mean to automatically compose these primitives? Finally, how can we fill the gap between a highly abstract primitive, which is by nature underspecified, and a concrete implementation, which needs to be fully functional?

4 CHALLENGE #2: CREATING INFRASTRUCTURE FOR PEOPLE AND AGENTS ON THE WEB

Functioning social machines require a combination of infrastructure, interactions that use the infrastructure, and people and agents to carry out those interactions. The process of creating infrastructure is time consuming, and extremely error prone; similarly, community building takes time, and relies heavily on network effects. These requirements are at odds with widening participation, requiring resources not available to the general public.

However, many innovations in Web engineering work by reusing existing infrastructure: “If This Then That”² provides simple scripts to work across multiple platforms; Zapier³ allows users to construct workflows based on web applications they already use; and work has been done on integrating electronic institutions with social media platforms [31]. This has the dual benefit of i) abstracting away authentication, security, storage provision etc. and ii) integrating with the practices that users already have.

This stands in contrast to typical development, which requires bespoke infrastructures to be hand-crafted by developers—clearly at the “expressive but difficult” end of the complexity spectrum. At the “accessible but constrained” end of the scale, Panoptes⁴ has made the deployment of crowdsourcing social machines so easy—in terms of specification, infrastructure and access to a community of volunteers—that several new projects are started each day.

Another observation is that dedicated infrastructure is actually not always needed: Social Machines can be built via social conventions, on top of existing communications infrastructure: a good example is the #IcanHazPDF social machine, which has created a community around a particular Twitter hashtag.

These social conventions go hand in hand with having a declarative modelling language, with specifications such as “reach consensus over X”, “discuss Y”. There are many ways these could be fulfilled that share a set of core executable semantics (i.e. providing a single answer that ‘most’ actors are comfortable with) while differing implementation details. Several threads of work are relevant here: “Do What I Mean” (DWIM [48]) allows abstract, intuitive, specifications and sensible error behaviours; convention over configuration supports extremely minimal configurations by providing sensible, compatible defaults for everything; finally, web service discovery techniques (e.g. [41]) are used to retrieve an appropriate web service implementation given a user-provided specification.

²https://ifttt.com/, see also [37]
³https://zapier.com
⁴https://panoptes.zooniverse.org/
The challenge then is whether existing platforms can be used as composable components in the design of social machines. A secure multi-agent voting protocol may be appropriate in some places, but sometimes a VOTE verb would be better served with a Doodle poll and bindings for computational agents to make use of the results. This requires a combination of both formal systems expertise and a sociological understanding. In particular, what range of human preferences and ad-hoc decisions can be intuitively described by an average user and is manageable automatically in terms of system configuration. We believe such a range would extend much further than a software system or web service.

5 CHALLENGE #3: ANALYSING AND DEBUGGING THE SOCIAL COMPUTER

As in traditional software engineering, we expect the design process for social machines to be cyclical: once designs are produced, they need to be evaluated and debugged, then updated accordingly. The designer should be able to analyse a model to validate that it accurately reflects their particular vision. It is also essential towards better understanding the functionality, limitations, and means of improvement of the social machine.

Agent-based simulation techniques [27] adapted to social machines can help test the flow and outcome of particular scenarios (e.g. a defined set of agents and parameters), in the spirit of unit tests. This would also enable explorative what-if scenarios with different parameters to gain insights on quantifiable properties (e.g. costs and delays), information flow, load balancing, etc.

Formal verification techniques, including model checking [9], can also be used to mathematically verify properties across all possible scenarios, ensuring the correct system behaviour, eliminate errors, and establish safety. They can also generate counterexamples of unwanted behaviour that breaks expressed properties, such as “the system never reveals information X about an agent”, value properties such as “if an agent pays for something then they will receive it or their money back”, and safety properties such as “no-one can steal money from another agent”.

The trade-off between expressiveness and automation is predominant here too. The logical languages employed by simulation and verification tools are seldom intuitive for the uninitiated, and therefore a more expressive, declarative language would be required. Executable semantics would also be required for such an analysis. These are available for languages such as LCC [42], BPEL (to some extent) and other protocol or workflow specification languages, but not in practical visual languages such as BPMN or flowcharts. Efforts to formalise the semantics of BPMN [13, 50] and develop formal verification tools [47] are clear indicators of the perceived usefulness of such techniques in the community.

Moreover, unlike purely technical systems, social machines include unpredictable human agents, and the overall “behaviour” of the machine depends both on the materiality of the system and on the collective agency of the user population. Testing and analysing this behaviour will require a realistic simulation of choices and social behaviour of human agents and of how the system’s regulative infrastructure affects the agents. This knowledge – which currently exists in mostly informal sources [25, 26, 51] – needs to be made available at the point of design.

Another key challenge of social machines is their adoption by the community [21, 40]. The social sciences provide a number of models to explain how and why humans engage with technology: these range from highly technical game-theoretic models [52] to empirical models from social psychology [25, 40]. The availability of such techniques requires modelling constructs describing both the technical elements (e.g. protocols) and social behaviour aspects (e.g. economic payoffs), making it a considerable challenge.

Finally, systematic monitoring a deployed social machine is the fundamental drive for refinement, continuous improvement, as well as investigation of the differences between the model and the actual social machine, particularly in the non machine mediated parts of the model. In addition, systems may change through use: new practices emerge which require infrastructure [6], or human behaviour is taken over by computational agents [33]. Related work has shown how social machine observation can be used to better understand social norms and incentives, and refine the interaction models to better support desired practice and optimise efficiency [32].

Such analysis requires automatic recording of event logs, perhaps in the form of provenance graphs [28, 30], and the use of techniques such as process mining [49]. One can then compare expected interaction models with the actual usage of the system (conformance checking), identify variances and exceptional behaviour, or even infer new interaction models. Such techniques have successfully been used within multi-agent systems [7, 44]. Making them usable by non-experts through intuitive interfaces and languages would be a major breakthrough towards our vision.

6 CONCLUSION

As social machines are rapidly becoming an integral part of today’s world and a frontier for the deployment of agent systems, it is paramount that their development becomes accessible to more than just large or well-funded institutions. Non-expert individuals should be able to design, build and analyse social machines that leverage distributed intelligence to benefit different communities and the general public. The challenges associated with this vision are as grand as its potential impact:

#1. How can we create intuitive models of complex interactions that balance what people want to express and what can be executed? How can we discover the primitive interactions that can become the building blocks of social machines? How can we incorporate intuitive but abstract, primitive but usable specifications?

#2. How can we make social machines easily deployable on the web? How can we leverage existing infrastructure and model-driven development to that end?

#3. How can we enable rigorous analysis and debugging that not only reveal system properties from the technical perspective, but also delve into the social aspects of these complex hybrid systems?

We argue that these questions require a unified approach involving research in multiple areas, including model-driven development, process modelling, agent-based simulation, game theoretic analysis, formal verification, software engineering and configuration, and social sciences. This is a unique opportunity to bring these research communities together, drawing strong contributions from them to bring the DIY social machine revolution within reach.