People-Oriented Programming: From Agent-Oriented Analysis to the Design of Interactive Systems

Steve Goschnick

Department of Information Systems
University of Melbourne, VIC 3010, Australia
stevenbg@unimelb.edu.au

Abstract. Where the Object-Oriented paradigm set about abstracting *objects*, Agent-Oriented (AO) theory draws on Psychology to abstract *mentalist notions* like: beliefs, perceptions, goals, and intentions. As such, the associated Agent-Oriented analysis can be used quite successfully to design interactive systems for people, delivering applications that are heavily individual-oriented. This reversal of the AO lens focuses analysis back upon people. It puts a multi-faceted agent used in analysis '*into the shoes*' of the user and turns the design and implementation into one we call *People-Oriented Programming (POP)*. POP calls on users to gather ethnographic data about themselves using Cultural Probes and on end-user innovation via software toolkits. This turn of focus is timely as the analyst/designer of interactive systems is facing new challenges regarding flexibility, user situatedness, dynamic environments, incomplete data, diversity in user needs, sensors in the environment, and users emersed in multiple parallel social worlds. Based on an extensive background analysis this paper distills a set of key aspects that any POP effort should possess.

Keywords: Agent-oriented analysis, agent-oriented paradigm, user innovation, HCI, people-oriented programming, agent meta-models, ShadowBoard Agents.

1 Introduction

Several Agent-Oriented (AO) architectures draw on models from Psychology (e.g. BDI and ShadowBoard [9]), abstracting *mentalistic notions*, such as: beliefs, perceptions, goals and intentions. As such, some associated agent-oriented analysis, can be used quite successfully to design interactive systems for individuals with heterogeneous needs. This reversing of the lens of AO back upon people, places a multi-faceted agent analysis '*into the shoes*' of the user and turns the design and implementation into one we call *People Oriented Programming*. This reversal of focus for AO analysis is timely, as modern interactive systems are placing new challenges upon the analyst/designer: a heightened degree of flexibility, situatedness of users, uncertain and dynamic environments, incomplete information, diversity in users and their needs, sensors proliferating in the environment, and users emersed in multiple parallel social worlds, instead of in one fixed organisation. It has a particular strength in the domestic setting, where people spend a significant amount of their time, often on non-work tasks, goals and less descript activities.

© Springer-Verlag Berlin Heidelberg 2009

Agent-Oriented analysis and design can deal with user *situatedness* via agent adaptability. An agent's internal *world view*, coupled with high-level *core value* goals [9] facilitates autonomous behaviour. An AO system can deal with the non-sequential external events in an agent's environment (*reactive behaviours*), while continuing with their current goals (*proactive behaviours*). In this paper we look in detail at these two aspects of interaction design in *mixed-initiative human-agent* systems [3]: dealing with the changing user context; and the *message-flow* model that facilitates reactive and proactive behaviours. It is presented in the context of the ShaMAN multi-agent meta-model [11], as instantiated in the DigitalFriend software [10].

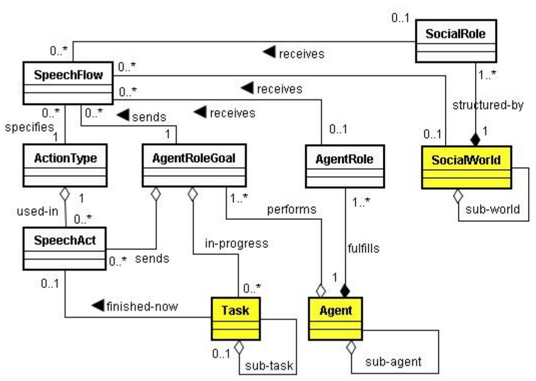
Then, as the main contribution, this paper presents *People Oriented Programming* (POP) as a new design paradigm for building personal systems. Based on an extensive background analysis it distills a set of key aspects that any POP effort should possess. POP calls upon the *user* in three capacities: as the focus of customised software, which von Hippel and Katz describe as '*markets of one*' [24]; as a self-ethnographer using Cultural Probes [7] to gather data; and as an end-user developer via software toolkits [24] designed to make the user central to innovation in new product development, the way that end-users are doing in the games genre [18] and in *mashups* of Internet services [2]. The technology used here to pursue *People Oriented Programming* is the DigitalFriend, V1 of which instantiates the ShadowBoard Agent Architecture [9]. Its theoretical base draws on Analytical Psychology – giving POP its fourth layer of meaning. Before looking at POP and the agent analysis, we first look at the pathways for *interaction* within the ShaMAN agent meta-model, in order to characterise and scope the design of the necessary interaction system.

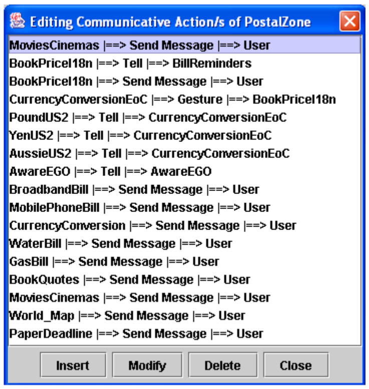
2 Interaction through Operators, SpeechActs and UI devices

The DigitalFriend V2 is multi-agent system (MAS) *software* – an instantiation of the ShaMAN meta-model – with a central goal of helping an individual user in the full-spectrum of their life (work, leisure, family, community activity, etc.). It is designed to monitor, alert, filter and initiate tasks, messages and resources, all within the context of the user's goals and activities. The Personal Assistant Agents accumulated within the DigitalFriend make it a *mixed-initiative* human-agent system [3]. Interactions that can take place, include: *inter-agent communication* which is facilitated via *speech-acts* in an agent communication language; *agent-to-user*, usually via messages accompanied by visual and/or aural alerts; and also via direct *user-to-agent interaction* usually accomplished through UI interface components.

2.1 Interaction through Speech-Acts

SpeechFlow in ShaMAN represents *an interaction model* for agent inter-communication. The allowable message types between two given agents, is an *interaction plan* (or a communication protocol), within which all the dynamically produced speech-acts, abide. Note: the right-hand side of figure 1 represents a part of an interaction plan, in the form of *sender* → *speechact* → *receiver*, for a number of sub-agents within a user's DigitalFriend.




Editing Communicative Action/s of PostalZone

MoviesCinemas |==> Send Message |==> User
BookPriceI18n |==> Tell |==> BillReminders
BookPriceI18n |==> Send Message |==> User
CurrencyConversionEoC |==> Gesture |==> BookPriceI18n
PoundUS2 |==> Tell |==> CurrencyConversionEoC
YenUS2 |==> Tell |==> CurrencyConversionEoC
AussieUS2 |==> Tell |==> CurrencyConversionEoC
AwareEGO |==> Tell |==> AwareEGO
BroadbandBill |==> Send Message |==> User
MobilePhoneBill |==> Send Message |==> User
CurrencyConversion |==> Send Message |==> User
WaterBill |==> Send Message |==> User
GasBill |==> Send Message |==> User
BookQuotes |==> Send Message |==> User
MoviesCinemas |==> Send Message |==> User
World_Map |==> Send Message |==> User
PaperDeadline |==> Send Message |==> User

Fig. 1. Message mapping and Agent Interaction Plans

On the left is the part of the ShaMAN meta-model that deals with the flow of speech-acts. The sending agent uses speech-acts to communication to either: other agents (including the user); to whole *SocialWorlds*; or to a specific *SocialRole* across *SocialWorlds*. In the DigitalFriend, each agent has a queue of messages received from the *SpeechFlow*, and the *situatedness* of the agent (particularly the *human user*), often determines which ones come off the queue, at what times, as we see in Section 3.

2.2 Human-to-Agent Interaction through UI Devices

The level of granularity of interaction between the human user and the agents in their DigitalFriend are varied. The concept of *Locales* is taken from Fitzpatrick [5], and can represent *any place* in which interaction takes place between the *members* of a Social World. This includes *abstract places* such as the graphical representation of a GUI on the screen. It can be something as commonplace as a File-Chooser GUI component that an agent uses to ask the user for a file. Or it can be a custom UI widget an agent uses to request specific information from the user.

Figure 2 is part of the ShaMAN meta-model as follows: an *Agent* has a number of roles in multiple *SocialWorlds* represented as *AgentRole*. Roles have a set of goals that once initiated, form an Agent's *intentions* (represented here as *AgentRoleGoal*). *Tasks* are set in motion to achieve a goal. Sub-tasks are performed by sub-agents, but some require the user to perform a task, or to ponder a new situation. An agent may call on *Human-Agent-Interaction* (the *HAI* entity) to achieve the necessary task. A Locale can be a place for interaction. I.e. the UI components in fig.3 include an *interactive map* of the world put to screen by an agent, which is waiting on the user to select and confirm a *country*. Where a traditional Task Analysis may go down to key-stroke interactions, in an agent system that uses high-level UI components, the task-granularity stops at *putting* the UI component to screen, and *receiving* the user's selection. I.e. The non-sequential nature of user interaction with a complex GUI component, is not of specific interest beyond what the user actually selects, facilitated here by the link between the *Task* and the *HAI* entities.

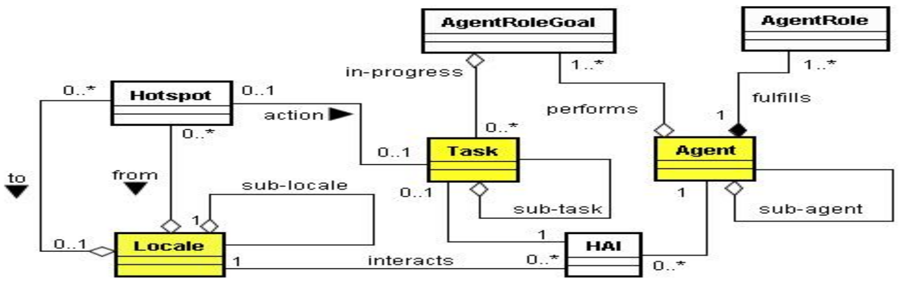
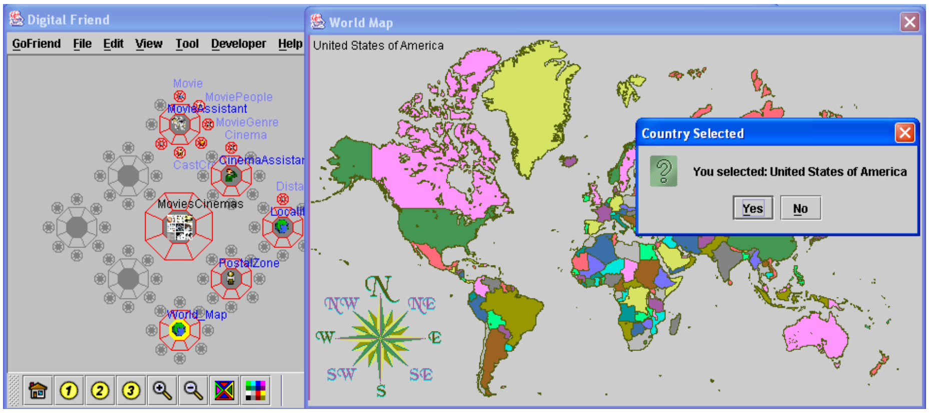


Fig. 2. Model of Human-Agent-Interaction within ShaMAN

3 Agent State as a Basis for Analysis

In the DigitalFriend the *user* is represented as another agent, but with several added features: agents can interact with the user via *direct actions* (in addition to speech-acts) via UI interfaces represented as Locales; the user has a set of known *behaviours* including *walking, driving, reading, meeting, sleeping*; and, the user is represented computationally at the top of the DigitalFriend's hierarchy of agents, with their Goals sitting at the top of the Goal hierarchy/network.



You selected: United States of America

Fig. 3. Human-Agent-Interaction within DigitalFriend V2

There is a complex representation of *agent-state* within the ShaMAN meta-model, giving a comprehensive coverage of the user's *situatedness* (figure 4), including what Locale they are currently in, who and what agents currently *inhabit* that Locale, what resources they have at hand, and more. In analysis it is paramount to discover the '*what*' rather than the '*how*' [17]. To do so, we assume all people in a user's Social-Worlds are equally empowered with a MAS of the sophistication of at least the DigitalFriend. In a given *Locale* the *Inhabitants* are assumed to be known, along with their *Roles*. Of the *Resources* across the system, those in *AgentResource* are immediately available to an *Agent*, in addition the agent may use *OnSiteResources*.

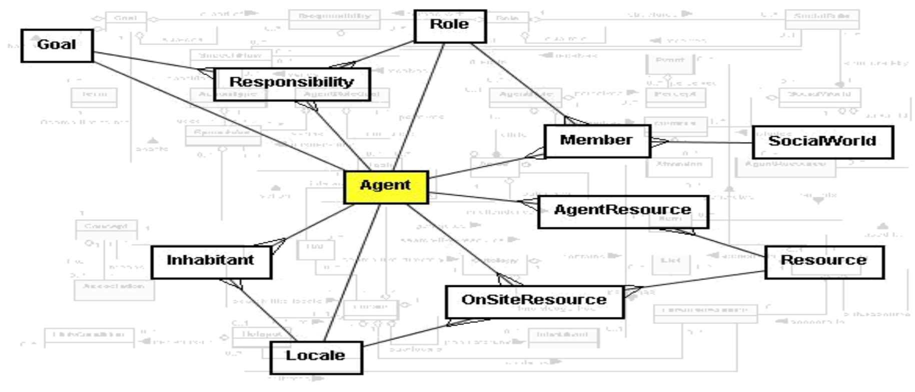


Fig. 4. Agent State in ShaMAN [11], with respect to situatedness

Note: The ShaMAN meta-model (the greyed-out background in fig. 4, available elsewhere [11]) has 30 classes/entities in UML class diagram notation. It effectively combines several sub-models: *Role*, *Goal*, *Task*, *Interaction, SocialWorld*, *Resource* and *Locale models*. These models are interconnected by a number of *associate entities,* in ER (entity relation) terminology – i.e. the entities on the *many* end of the *one-to-many* relationships, in crows-foot notation. E.g. *Inhabitant, Responsibility, Member, AgentResource* and *OnSiteResource* cross-relate several sub-models, to great benefit regarding *situatedness,* in analysis, design and implementation.

Communication to the user from the DigitalFriend can be filtered for their current *situation*. These elements of state (as per figure 4 above), together with the user's *current* behaviour, allows ShaMAN to select messages from the agent's current queue of messages (in particular, *the user*), according to a set of *message delivery rules*.

It becomes easy to stipulate rules such as: *No taking a phone call while driving*. The rules can be stipulated more clearly once a user's state has been ascertained: What SocialWorlds do they belong to, and what are their Roles in those worlds? What are the Responsibilities that go with those Roles? Who else are members of those SocialWorlds and what are their respective Roles? What Locales (real and abstract) does the user frequent in the course of fulfilling their Responsibilities? What are the current conditions in the Locale? What Resources does the user have at their disposal, within each Role? What Resources do their agents have at their disposal? What Resources are available in the Locales a user is expected to use? What Agents does the user expect to have available when in each of the Locales? What Agents does the user expect to have available when they are in each of their Roles?

4 People-Oriented Programming as a Design Paradigm

4.1 Background

This general method follows on from the Shadowboard *Methodology* by Goschnick and Graham [12]. It was not a generic AO methodology, but was specifically aimed at gathering and tailoring the AO *requirements* for a multi-agent Personal Assistant Agent (PAA) system, for an individual user. The authors made the point with:

"…the primary requirement in the work presented here was to have a 24x7 user-representation available (even) while the user sleeps or is otherwise offline, within a tool capable of autonomous computation, some decision-making, some information filtering and with the ability to concentrate the presentation of relevant information and to inform the user at the most convenient time"

Their central idea then was to marry a top-down 62 role/sub-role model starting template, with certain bottom-up techniques from ethnography including Cultural Probes by Gaver et al [7] in the form of "*user-kept diaries"* and user *scenarios* in the tradition of Rosson and Caroll [19], into a methodology that addressed the *personal aspirations and desires of an individual* together with their *social needs*. To bring an individual's social needs into the equation, Goschnick and Graham foreshadowed future work that called upon the theory of Social Worlds by Strauss [21] and the Locales framework [5]. That foreshadowed work there, is the ShaMAN meta-model here, which now facilitates Social Worlds and Locales in a MAS system.

A significant difference in the *Shadowboard Methodology* over other agent methodologies involves the Role entity: a *role model* is not only used in the *requirements gathering* process, in addition, it serves functions within the analysis, design and in the *implementation* of a Personal Assistant Agent. E.g. A *role-hierarchy lens* is used to filter and organise messages for the user's benefit, that come from various sub-agents [10]. These messages are also stored in a log that represents an *interaction trajectory arc* [5] of the user's life so far.

Cultural Probe [7] data captures the richness of individual users in the domestic and social space, however, many Ethnographers use it to *inspire* their own design processes [1,8] rather than to enhance the design process. There have been pockets of research trying to bring ethnographic data and/or scenarios into the Software Engineering process as seamlessly as possible [14,22,6,13]. Some provide support for a multi-disciplinary team approach to bring in the richness of social context [13]. While ethnography and software engineering are complementary - ethnographic studies capture the *details* that are useful in analysis, while software engineering design looks for and uses abstractions as often as possible [22] - the two forms of data and focus, from these two quite different disciplines, each with different notations and concepts, means that there has been a bottleneck in getting from one to the other, difficult to negotiate without loosing much of the detail captured in the ethnographic data. Furthermore, there is neither an efficient or affordable way to capture it *on the scale needed* for *heterogeneous individual user needs*.

A way forward involves users collecting data about their own lives with Cultural Probes and Software Engineers providing *model-based toolkits* to enable end-user development of interfaces and mash-ups of the software and Internet-based services they regularly use in their lives. There is a movement of people towards such end-user development of software and computer-based artifacts – whenever they get the appropriate tools to do so. We can see it in the current rise of mash-ups in the Internet world [2] by hobbyist users. And we see it in the user modifications (so-called *mods*) in computer games that facilitate user additions to their game playing [18, 16].

End-user development is in part researched in the context of *user innovaton* – i.e. innovations created by end-users themselves rather than by corporate software houses [23]. Von Hippel and Katz investigated the use of toolkits [24] distributed to end-users, in order for manufacturers to be able to service the unique needs of individuals in what they called "*markets of one*". I.e. Some manufacturers have abandoned their "*increasingly frustrating efforts to understand users' needs*" (ibid) and instead have outsourced need-related innovation tasks to the users themselves. To do so, the tasks involved in bringing a new product into existence, are divided into two interrelated parts: *solution-related* tasks, and *needs-related* tasks. The solutions-related tasks are catered for with flexible user-friendly toolkits, initially provided by the manufacturer. The *needs-related* tasks are what the end-users then do with those toolkits – i.e. they customise the initial product to suit their specific needs.

Von Hippel has been researching in the user innovation space since the early 1990's and in his book Democratizing Innovation [23] he gives two primary reasons that help to explain the recent exponential growth in the user-innovation area: tools that were previously only available to a niche professional base, have become available to mass end-users (in both cost and ease-of-use); and secondly, online communities of end-users confide their needs and share their solutions through the various communication channels and social networks afforded by the Internet.

One of the earliest mass-enlistments of end-users via software toolkits, is in the computer games area [15], where numerous games have tens of millions of users, and several of them have tens of thousands of end-users providing additional innovative content and functionality to those games. Prugl and Schreier [18] studied the use of toolkits for the popular computer game *The Sims* (28 million units sold within 2 years of release), in which they studied samples from four online user communities, with an average of 15,000 members (ibid). Many other games offer toolkits to end-users to extend game functionality and content. Jeppesen and Molin [15] found that of the 94 games they surveyed, 33 of them included toolkits for end-user development.

Where von Hippel mainly describes end-user innovation as the way that 'markets of one' can be appropriately and cost effectively serviced with the goal of '*satisfying each user's needs*', Prugl and Schreier looked deeper into '*how*' users deal with the invitation to innovate (including the *model of open innovation*), and they also investigated the attractiveness of end-user designs, to other users. They single out '*leading-edge*' users as a potential source of radical product development (ibid), since their designs find large user-bases amongst other users in online communities that centre around the toolkits. This is a useful finding since end-user toolkits are used by *a minority* of users, whereas some of the innovation produced by those users, can be used by many of the rest. In a study about *what motivates* users to modify the games they play [16], Kadarusman focused on the *World of Warcraft* (WoW) game, which has over 11 million registered users (as at October 2008). He reported that WoW has more than 4600 user-modifications available for download, the most popular of which was downloaded on average 110,927 times per day.

4.2 Definition of People-Oriented Programming

We are now in a position to describe *People Oriented Programming* (POP) as a new design paradigm for developing individual-oriented systems, and define the four elements that it includes. POP calls upon *the individual user* in three main capacities: firstly, as the *central focus* of a customised software system addressing heterogeneous needs, which von Hippel and Katz describe as '*markets of one*' [24]; secondly as a self-ethnographer [12,1] administering and using Cultural Probes [7], personal role models [9,10] and scenarios [19] to gather their own very-specific data (including in the domestic space); and thirdly, as end-user developers, coming up with their own solutions to match their personal needs, utilising well-engineered software toolkits [24] designed to make the user the centre of innovation in new product development. The fourth element of POP is the *cognitive models* behind the tools, techniques and frameworks upon which the user toolkits are built. These models are drawn from two disciplines that are not often cross-referenced [11]: the *Agent-Oriented paradigm*, and *Cognitive Task Modelling*. E.g. the technology used in this research to pursue *People Oriented Programming* is the DigitalFriend, which appropriately has its theoretical base anchored upon a century of evolution of *models of mind* from Analytical Psychology [9,10].

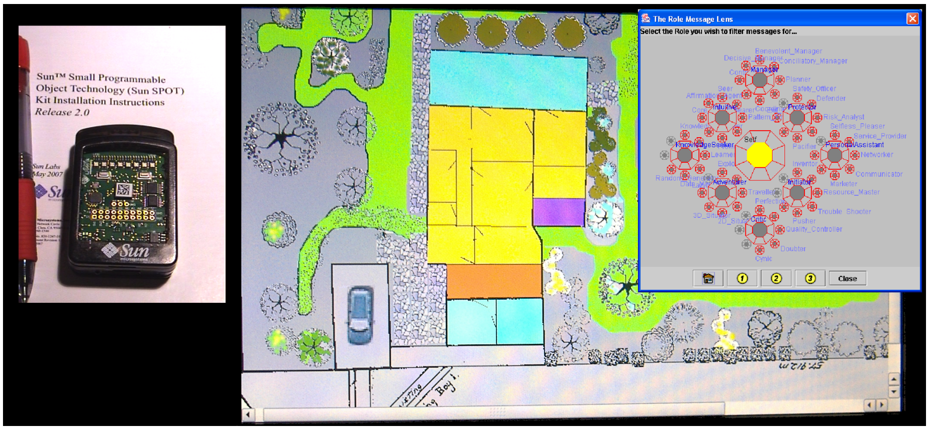

Sun™ Small Programmable Object Technology (Sun SPOT) Kit Installation Instructions Release 2.0
The Role Message Lens
Select the Role you wish to filter messages for...
Manager
Protector
Intuitive
Knowledge Seeker
Self
Personal Assistant
Adventurer
Initiator
Critic
Close

Fig. 5. Home Environment Locale with interface to Java SunSpot sensor (insert)

The following section briefly explores the use of *Locales* from the ShaMAN meta-model, which has featured in parts of Sections 2 and 3, by way of an example DigitalFriend of a user engaged in POP, with the DigitalFriend V2 toolkit:

4.3 Example Locales in a User's DigitalFriend

In the personalisation of the DigitalFriend for a given individual, numerous private and personally significant *Locales* are brought into the analysis, and into the technology. For example, a map of the user's home is represented as a Locale in figure 5. It is cross-related with the *Role lens* in the DigitalFriend (see *insert* in figure 5), and is connected with *a sensor* (a Java *SunSpot* technology kit, in this prototype) that knows exactly where the user currently is spatially. For example, if the person is in the kitchen sub-Locale, then the DigitalFriend can be set to assume the person is in the default *role* of *cook*, and likewise the Resources that become available can be cooking suggestions and recipes. When in the garden, they can be assumed to be in the *gardener* role by default, and can receive messages about their previous activity against significant Resources (e.g. "*You last pruned the Apple tree in late Winter of 2008*") from a *trajectory arc* (i.e. via log files). With more specialised sensors, they would be able to get information about the state of the garden as they pass by them – e.g. moist content sensors could trigger: "*The south-east garden bed has a moisture content well below the recommendation for this time of the year*".

The Locales in the model do not have to represent maps or rooms that actually exist, as those in figure 5 do. From the Jungian Psychology that underlines the theory of sub-selves behind the many-facetted, many-role model of the *individual*, also comes the Jungian concept of *archetypal* symbols [9], which hold common meanings (across people) when they appear in peoples' dreams. From Jung we are told that to dream often of a given House or Home usually symbolically represents the person's mind itself in some compartmentised way. The different rooms: *kitchen, lounge, dining, bedroom, laundry*, etc symbolically representing analogous-aspects of the individual's life. Therefore an individual user can build '*the house of their dreams*' in computer-based imagery, either 2D plans or 3D representations, and then link those images/media to the roles in their life within the MAS DigitalFriend, as a personally satisfying and highly intuitive interface.

5 Conclusion

People Oriented Programming (POP) as defined above sounds simple i.e.: focus on the heterogeneous needs of individual users; get the users to record their own ethnographic data; and then have them develop their own enhancements to agent-oriented software using user-friendly toolkits. That stated simplicity belies the actual complexity to carry it out. Just as the modern GUI PC *is much easier* to use than old non-GUI PCs, multiple layers of complexity and indirection were needed to bring about that simplicity of use. Not surprisingly then, ethnography is an inexact way to gather requirements as compared to traditional requirements engineering methods; AO technologies are an order of magnitude more complex that traditional OO languages and frameworks; and user-oriented toolkits that are user-friendly enough to build personal systems from disparate services and applications, are complex in terms of designing and building them. However, the POP approach retains the richness from the cultural probe data, through into the technology in a way that reflects peoples social needs, desires and goals, and to the benefit of the collective aspirations of the *social worlds* they are a part of.

The recently reported amount of *end-user innovation* in the games genre touched on above, is testimony to the approach working, when the mix between *user-needs* and the functionality of the *solution-related* technology on offer, is right.

The *agent-oriented model-based* approach to personalising an individual's interface to the technology in their lives is a natural fit. The mentalistic notions that the AO paradigm abstracts in a computational form, draws upon Psychology, and therefore can be reverse-focused upon programming for individuals, by end-users themselves. The models from *cognitive task modeling* told us that *goals, plans, tasks, actions, roles* and *objects* are represented in the cognitive functioning of the mind. These models confirm those from the *agent-oriented paradigm* [11], and vice versa, through their strong underlying similarities. And it is the *models* that will hold POP together as the artifacts shared between the Software Engineers building the toolkits, and the end-users innovating their own creations and customisations with them.

References

1. Arnold, M.: The Connected Home: Probing the Effects and Affects of Domesticated ICTs. In: Proceedings of PDC 2004, ACM, Toronto (2004)
2. Feiler, J.: Web 2.0 Mashups. McGraw Hill, New York (2008)

3. Fleming, M., Cohen, R.A.: User Modeling Approach to Determining System Initiative in Mixed Initiative AI Systems. In: International Conference on User Modeling UM 1997 (1997)
4. Fitzpatrick, G.: The Locales Framework: Understanding and Designing for Wicked Problems. Kluwer Academic Publications, London (2003)
5. Forbrig, P., Dittmar, A.: Bridging the Gap between Scenarios and Formal Models. In: Jacko, J., Stephanidis, C. (eds.) Human Computer Interaction: Theory & Practice (Part I), pp. 98–102. Lawrence Erlbaum Associates, Mahwah (2003)
6. Gaver, B., Dunne, T., Pacenti, E.: Cultural Probes. Interactions 6(1), 21–29 (1999)
7. Gaver, B., Boucher, A., Pennington, S., Walker, B.: Cultural Probes and the Value of Uncertainty. Interactions 11(5), 53–56 (2004)
8. Goschnick, S.B.: ShadowBoard: an Agent Architecture for enabling a sophisticated Digital Self. Thesis, Dept. of Computer Science, University of Melbourne, Australia (2001)
9. Goschnick, S.B.: The DigitalFriend: the First End-User Oriented Multi-Agent System. In: OSDC 2006, Open Source Developers' Conference, Melbourne, Australia, December 5-8 (2006)
10. Goschnick, S., Balbo, S., Sonenberg, L.: From Task to Agent-Oriented Meta-models, and Back Again. In: Forbrig, P., Paternò, F. (eds.) HCSE/TAMODIA 2008. LNCS, vol. 5247, pp. 41–57. Springer, Heidelberg (2008)
11. Goschnick, S., Graham, C.: Augmenting Interaction and Cognition using Agent Architectures and Technology Inspired by Psychology and Social Worlds. In: Universal Access in the Information Society, vol. 4(19), Springer, Heidelberg (2005)
12. Haesen, M., Coninx, K., Van den Bergh, J., Luyten, K.: MuiCSer: A Process Framework for Multi-disciplinary User-Centred Software Engineering Processes. In: Forbrig, P., Paternò, F. (eds.) HCSE/TAMODIA 2008. LNCS, vol. 5247, pp. 150–165. Springer, Heidelberg (2008)
13. Hughes, J., O'Brien, J., Rouncefield, M., Blythin, S.: Designing with Ethnography: A Presentation Framework for Design. In: Proceedings of Design of Interactive Systems (DIS 1997), pp. 147–158. ACM, Amsterdam (1997)
14. Jeppesen, L.B., Molin, M.J.: Consumers as Co-developers: Learning and Innovation Outside the Firm. Technology Analysis and Strategic Management 15(3), 363–384 (2003)
15. Kadarusman, J.: User-Innovation in the Modding Community of World of Warcraft. Honours Thesis, Department of Information Systems, University of Melbourne (2008)
16. Pressman, R.: Software Engineering: A Practitioner's Approach. McGraw-Hill, New York (2004)
17. Prugl, R., Schreier, M.: Learning from leading-edge customers at The Sims: Opening up the innovation process using toolkits. R&D Management 36(3), 237–250 (2006)
18. Rosson, M.B., Caroll, J.M.: Usability Engineering: Scenario-based development of human-computer interaction. Morgan Kaufmann, San Francisco (2001)
19. Stary, C.: Toward the Task-Complete Development of Activity-Oriented User Interfaces. International Journal of Human-Computer Interaction 11(2), 153–182 (1999)
20. Strauss, A.: A Social World Perspective. Studies in Symbolic Interaction 1, 119–128 (1978)
21. Viller, S., Sommerville, I.: Conherence: An Approach to Representing Ethnographic Analyses in Systems Design. Human-Computer Interaction 14, 9–41 (1999)
22. Von Hippel, E.: Democratizing Innovation. MIT Press, Cambridge (2005)
23. Von Hippel, E., Katz, R.: Shifting Innovation to Users via Toolkits. Management Science 48(7), 821–833 (2002), `http://userinnovation.mit.edu/papers/10.pdf`