Personalization and Personification: A Constructive Approach Based on Parametric Agents

Fabrice Popineau, Georges Dubus, and Yolaine Bourda

SUPELEC Systems Sciences (E3S),
Computer Science Department,
Gif-sur-Yvette, France
{fabrice.popineau,georges.dubus,yolaine.bourda}@supelec.fr

Abstract. In this paper, we illustrate PAGE – for Parametric AGEnts, a constructive way to design personalized or personified virtual agents starting from a plain rational agent. We chose a scenario about the personification of a virtual agent in a serious game for training communication skills. We start from an agent program that does not take into account any kind of agent profile. We show how our PAGE approach allows us to derive different agent programs tailored to different agent profiles while retaining complete control on the resulting agent behavior.

1 Scenario

We address a scenario inspired from [1] and [7]. In a serious game for training communication skills, a player plays the role of a real estate agent to convince a non-player character, played by a virtual agent, to visit a property. During the discussion, the player can ask questions to discover the wishes of the IVA and can communicate information, interpretations and opinions in order to convince the IVA that the property meets its expectations, by highlighting the qualities of the property that echoes the wishes of the buyer. The IVA may also ask questions to get the information he wants, give opinions, and make the decision to visit the property or terminate the conversation.

The PAGE framework is an enhanced and refined version of the work presented in [4]. Two traits of the IVA personality are modeled: the *extraversion* and the *agreeableness* of the Five Factor model [6]. These features are modeled at a coarse grain, without considering for example the facets and schemes of the NEO PI-R model [3].

In [7], the IVA is driven by a BDI agent whose beliefs are the knowledge the IVA has on the property, whose goals are to obtain information and to make a decision, and whose plans are conversation strategies to achieve the goals. The pseudocode in listing 1 illustrates how the personality of the IVA is implemented: the behaviors are given in the form of conditional expressions depending on the psychological profile of the agent. This approach suffers from the multicriterion bloat: the complexity grows exponentially with the number of traits.

T. Bickmore et al. (Eds.): IVA 2014, LNAI 8637, pp. 339–344, 2014.

Listing 1. Pseudocode for selecting the type of information

```
if (agreeable) then                  /* Boolean                          */
  if (extrovert) then                /* Boolean                          */
    if (probability > .5) then       /* With equal chance:               */
      return: tell(wish)             /* Tell a wish                      */
    else
      return: tell(opinion)          /* Tell an opinion                  */
  else                               /* Act introvert:                   */
    return: tell(fact)               /* Tell fact about the house        */
else                                 /* Act non-agreeable:               */
  return: tell(fact)                 /* Tell fact about itself           */
```

The evaluations published in [1] and [7] about this scenario show the usefulness of the personification of the agent. An agent expressed with BDI can be translated into an agent expressed in Golog while keeping the same architecture as shown in [8], but the multicriterion bloat complexity will be inherited. Our PAGE approach solves this problem by giving an appropriate transformation of the program for a given profile.

2 PAGE Process

The PAGE process has been enhanced on several points since [4]. This process is based on a set of transformations that are recursively applied to the original Golog program. The transformation process is driven by the profile we consider. The application designer provides a connection table which relies the traits in the profile to actions that are related to these traits. The first goal of these transformations is to allow to prefer actions which are more geared towards the profile. The notion of *families of actions* and *families of procedure* have been refined to open up the set of available ways to achieve a given result. In addition to preferences (denoted by *pref+* and *pref-* later on), we also have introduced the ability to block or force the use of certain ways to achieve results: some behaviors may be required (*req*) or banned (*ban*). Last, actively sensing user actions is essential for an agent interacting with humans and requires online planning ability. Our transformation process can transform programs as a whole, hence is not subject to the sensing barrier contrarily to other approaches like DT-Golog [2] [9] [5].

This gives us a rich set of operations that enables us to transform a profile independent Golog program into one which is either personalized or personified. We detail an example of the latter below.

proc *shareInformation*
  shareWish
  | *shareOpinion*
  | *shareFactAboutHouse*
  | *shareFactAboutItself*
endProc

(1) **proc** *evaluate*(*message*)
  $\pi f; message = tell(f)?;$
  if $f = Fact(\_)$
  then $assert(f);\ ackApprove(f)$
  else $\neg incompatible(f)?;\ assert(f); ackApprove(f)$
    | $ackReject(f)$
  endIf
endProc (2)

3 Agent Modification

The buyer agent can be personified in many ways. We show two of them, and show that they interact seamlessly. Transformations will be applied on procedures 1 and 2.

Selecting the type of information to share. The agent can choose to share information with the player (eq:shareInformation). The standard agent will choose randomly between *shareWish*, *shareOpinion*, *shareFactAboutHouse* and *shareFactAboutItself*. An *extrovert* character will tend to express his wishes and opinions, while an introvert will express facts and ask questions. An *agreeable* character will tend to ask about the house or the environment, while a disagreeable character will tend to talk about himself. In the implementation of [1], this personalization is done by enumeration of the possibilities (listing 1).

We express *introvert* as the negation of *extrovert* ($neg(extrovert)$). That way, the introvert agent will avoid actions associated with extrovert, and vice versa. Given the description of the traits *agreeable* and *extrovert*, we define the function *attributes*:

$attributes(shareWish) = attributes(shareOpinion) = \{agreeable, extrovert\}$
$attributes(shareFactAboutHouse) = \{agreeable, neg(extrovert)\}$
$attributes(shareFactAboutItself) = \{neg(agreeable), neg(extrovert)\}$

The definition of *attributes* enables the automatic transformation by the PAGE process of the *shareInformation* procedure according to a profile. The results for various profiles are given in table 1 where the names have been shortened (the prefix *share* has been removed). This approach gives us variations in programs, and thus let us define complex behaviors without having to state the result of each trait combination.

Table 1. Transformation of the *shareInformation* procedure under different profiles

agreeable	*extrovert*	Transformed procedure
pref-	*pref-*	*shareFactAboutItself* ⟩ *shareFactAboutHouse* ⟩ (*shareWish* | *shareOpinion*)
pref-	*neutral*	*shareFactAboutItself* ⟩ (*shareWish* | *shareOpinion* | *shareFactAboutHouse*)
pref-	*pref+*	(*shareWish* | *shareOpinion* | *shareFactAboutItself*) ⟩ *shareFactAboutHouse*
neutral	*pref-*	(*shareFactAboutHouse* | *shareFactAboutItself*) ⟩ (*shareWish* | *shareOpinion*)
neutral	*neutral*	*shareWish* | *shareOpinion* | *shareFactAboutHouse* | *shareFactAboutItself*
neutral	*pref+*	(*shareWish* | *shareOpinion*) ⟩ (*shareFactAboutHouse* | *shareFactAboutItself*)
pref+	*pref-*	*shareFactAboutHouse* ⟩ (*shareWish* | *shareOpinion* | *shareFactAboutItself*)
pref+	*neutral*	(*shareWish* | *shareOpinion* | *shareFactAboutHouse*) ⟩ *shareFactAboutItself*
pref+	*pref+*	(*shareWish* | *shareOpinion*) ⟩ *shareFactAboutHouse* ⟩ *shareFactAboutItself*

Accepting interpretations. When the player gives the agent information, the agent can accept or reject it. Facts are always accepted, because the agent consider the player doesn't lie. Opinions and interpretations may be rejected if they contradict the knowledge base. This behaviour is defined in the procedure *evaluate* (2).

To personify the agent, we extend the basic behavior of *ackApprove()* and *ackReject()* (acknowledge approval or rejection in a neutral manner) by defining

two families of procedures. First, the procedure *ackApproveWithOpinion* (3) lets the agent acknowledge approval while giving his opinion. It is only available if the agent has an opinion on the subject. It is associated with an *extrovert* behaviour. The second procedure *ackApproveSilently* (8), lets the agent say nothing after accepting an information. A third procedure *ackRejectSilently* (9) lets the agent reject an information without saying anything. Both procedures are different because they are part of different families: the first one can be used when accepting a piece of information, the other one when rejecting. Both are associated with a *disagreeable* behaviour. Two families (10 and11) state what procedures can be used instead of others. Finaly, we state (7) that adding a message to the knowledge base is associated to the trait *agreeable*. The PAGE process uses these families to replace *ackApprov* and *ackReject* by other procedures that are more suitable for the profile.

For an extrovert agent ($\{extrovert : pref+, agreeable : neutral\}$), the result of the transformation is procedure 12. The differences with (2) are underlined. For each family of procedures, (*ackApprove* and *ackReject*), each member of the family is ranked with regard to the profile. Thus *ackApproveWithOpinion* is of class *pref+*, since its unique attribute is *extrovert* with value *pref+* in the profile. Similarly, *ackRejectSilently* and *ackApproveSilently* are of class *pref-* and *ackApprove* and *ackReject* have no class. Procedures of negative classes are dismissed, and the others are replaced by a choice between the remaining procedures in families with a preference for procedures of the highest rank: $ackApprove(f)$ is replaced by $(ackApproveWithOpinion(f)) \rangle ackApprove(f)$. This is the only change that has an effect in this example. For a disagreeable agent($\{extrovert : neutral, agreeable : pref\text{-}\}$), the result is procedure 13.

Combining transformations. Both transformations apply to different parts of the program, hence it is possible to apply them simultaneously to get an agent having both alterations of his behavior. Problems only arise in cases where alterations deal with the same parts of the program, in which case the transformation process may give several results. For this reason, we believe that the process has to stay semi-automatic: the designer of the application may want to validate the result of the transformation.

proc *ackApproveWithOpinion*(f) (3)
  $\pi predicate, opinion$
  $f = \_(predicate, \_)$
    $\wedge\, KB(Opinion(predicate, opinion))$?
  if $opinion > 0.5$ **then**
    $acknowledge(accept\_happy)$
  else
    $acknowledge(accept\_unhappy)$
  endIf
endProc

$attributes(ackApproveWithOpinion) = \{extrovert\}$ (4)

$attributes(ackApproveSilently) = \{neg(agreeable)\}$ (5)

$attributes(ackRejectSilently) = \{neg(agreeable)\}$ (6)

$attributes(assert(f)) = \{agreeable\}$ (7)

```
proc ackApproveSilently(f)                    (8)
   nil
endProc
proc ackRejectSilently(f)                     (9)
   nil
endProc
```

```
family({ackApprove,                           (10)
        ackApproveWithOpinion,
        ackApproveSilently})
family({ackReject,                            (11)
        ackRejectSilently})
```

```
proc evaluate(message)                        (12)
   πf; message = tell(f)?;
   if f = Fact(_) then
      assert(f);
         (ackApproveWithOpinion(f)
         ⟩ackApprove(f))
    else
      ¬incompatible(f)?; assert(f);
         (ackApproveWithOpinion(f)
         ⟩ackApprove(f))
       | ackReject(f)
    endIf
endProc
```

```
proc evaluate(message)                        (13)
   πf; message = tell(f)?;
   if f = Fact(_) then
      assert(f);
         (ackApproveSilently(f)
         ⟩ackApprove(f))
    else
         ackRejectSilently(f)⟩ackReject(f)
      ⟩¬incompatible(f)?; assert(f);
         (ackApproveSilently(f)⟩ackApprove(f))
    endIf
endProc
```

4 Conclusion

We have illustrated how to use our PAGE approach on a real-case scenario which has been previously assessed. We have shown that our approach allows for a simpler and clearer expression of the behavioral change than a classical approach. Our next step is to provide an interactive tool so that the agent designer can chose what transformations to apply at each point in the program. Families of actions have also to be investigated from an ontological point of view. Last, we have considered personalization and personification as exclusive transformations, but we will find extremely interesting to address the problem of giving both behaviors to an agent program.

References

1. Van den Bosch, K., Brandenburgh, A., Muller, T.J., Heuvelink, A.: Characters with personality! In: Nakano, Y., Neff, M., Paiva, A., Walker, M. (eds.) IVA 2012. LNCS, vol. 7502, pp. 426–439. Springer, Heidelberg (2012)
2. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S., et al.: Decision-theoretic, high-level agent programming in the situation calculus. In: AAAI/IAAI, pp. 355–362 (2000)
3. Costa, P.T., McCrae, R.R.: The NEO PI-R professional manual. Psychological Assessment Resources, Odessa (1992)
4. Dubus, G., Popineau, F., Bourda, Y., Sansonnet, J.P.: Parametric reasoning agents extend the control over standard behaviors. In: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) (2013)

5. Ferrein, A., Fritz, C., Lakemeyer, G.: Using golog for deliberation and team coordination in robotic soccer. KI 19(1), 24 (2005)
6. Goldberg, L.R.: An alternative description of personality: The Big-Five factor structure. Journal of Personality and Social Psychology 59, 1216–1229 (1990)
7. Muller, T.J., Heuvelink, A., van den Bosch, K., Swartjes, I., et al.: Glengarry glen ross: Using bdi for sales game dialogues. In: AIIDE (2012)
8. Sardina, S., Lespérance, Y.: Golog speaks the BDI language. In: Braubach, L., Briot, J.-P., Thangarajah, J. (eds.) ProMAS 2009. LNCS, vol. 5919, pp. 82–99. Springer, Heidelberg (2010)
9. Soutchanski, M.: An on-line decision-theoretic golog interpreter. In: IJCAI, pp. 19–26. Citeseer (2001)