Xipho: Extending Tropos to Engineer Context-Aware Personal Agents

Pradeep K. Murukannaiah
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
pmuruka@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

ABSTRACT

We introduce Xipho, an agent-oriented methodology for engineering context-aware personal agents (CPAs). Xipho extends Tropos to support CPA development. Xipho's steps span a CPA's requirements acquisition, design, and implementation. Importantly, we treat context as a cognitive notion and systematically relate it to other cognitive notions such as goals and plans. Xipho incorporates reusable components in a CPA's design and implementation to simplify the development process. We evaluate Xipho empirically, finding that Xipho reduces development time and effort, and improves the comprehensibility of CPA designs.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements and Specifications—*Methodologies*

General Terms

Design, Experimentation, Human Factors

Keywords

Methodology, Context, Personal agent, Tropos

1. INTRODUCTION

Humans have an inherent understanding of the contexts in which they act and interact. A *context-aware personal agent* (CPA) adapts to the contexts of its human user. CPAs can prove valuable in settings such as healthcare, smart (physical or virtual) environments, and e-commerce.

Engineering a CPA is nontrivial. First, a CPA must capture its users' mental models of context—a high-level concern centered on meaning. Second, a CPA must acquire the desired contextual information—a low-level concern centered on devices and infrastructure. We describe Xipho, a methodology for systematically developing a CPA.

We treat context as a cognitive notion and understand other cognitive notions, such as goals and plans, as inherently related to context. Thus, a natural approach to developing a CPA would be an agent-oriented software engineering (AOSE) methodology that employs cognitive notions throughout development. Existing AOSE methodologies, e.g., [6, 17, 22], describe generic steps of software development, but fall short in dealing with challenges specific to CPA development. Xipho fills this gap by providing systematic steps for (i) capturing a CPA's contextual requirements, (ii) deriving a context information model specific to a CPA, and (iii) leveraging reusable components in a CPA's design and implementation. These tasks pose nontrivial challenges (described below). We demonstrate that Xipho addresses the challenges of CPA development successfully.

A CPA's contextual requirements are often (i) unknown ahead of time, and (ii) subject to change as users employ the CPA in different contexts. A developer, inevitably, makes assumptions about a CPA's users' contextual needs, but often buries such assumptions in the implementation. Xipho provides constructs necessary to explicate such assumptions, yielding easily renewable CPAs [20].

Context is traditionally defined as "any information relevant to an interaction between a user and an application" [8]. Accordingly, the space of what constitutes context is vast, e.g., a user's location, activities, emotions, or social setting. Existing techniques [4] model context as a notion generic across applications and users, which a developer must tailor to the requirements of a CPA. Xipho assists a developer in restricting a CPA's context information model to a set of abstractions meaningful in the CPA's scope, enabling the CPA to offer a natural user experience.

To acquire the desired context information, a CPA must address device and network centric concerns such as sensing, aggregating, and propagating context information. However, the concerns of context acquisition can be separated from those of CPA development. Importantly, several existing works address the problem of context acquisition [2]. Xipho enables a developer to build a CPA on reusable components, reducing the cognitive burden of CPA development and yielding easily comprehensible CPA designs.

Contributions. Our contributions are two fold. First, we propose Xipho as an extension of Tropos for developing CPAs. Xipho addresses nontrivial challenges associated with requirements acquisition, design, and implementation of a CPA. We demonstrate Xipho via a case study, which involves the engineering of a smart phone based CPA.

Second, we evaluate Xipho through a study in which 46 developers applied Xipho to engineer three CPAs. Its results support our claims that Xipho (i) reduces the time and effort required to develop a CPA, and (ii) yields CPA designs that are easy for other developers to comprehend.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. BACKGROUND: TROPOS

Xipho extends Tropos [6] to support CPA development. Specifically, Xipho adopts the Tropos metamodel, which consists of the following main constructs.

Actor: A social, physical, or software agent (or a *role* of an agent). An actor has goals within a system.

Goal: A strategic interest of an actor. A *hard goal* has a crisp satisfactory condition. A *soft goal* has no clear-cut definition or criterion for deciding whether it is satisfied or not. Thus hard goals are *satisfied* whereas soft goals are *satisficed*.

Plan: An abstraction of doing something. Executing a plan is a means of satisfying or satisficing a goal.

Resource: A physical or information entity.

Dependency: A relationship indicating that a *depender* actor depends on a *dependee* actor to accomplish a goal, execute a plan, or furnish a resource. The object (goal, plan, or resource) here is the *dependum*.

Belief: An actor's representation of the world.

Capability: An actor's ability to choose and execute a plan to fulfill a goal, given certain beliefs and resources.

Our motivation to extend Tropos is that it spans all development phases. First, Tropos captures *early* and *late* requirements as *system-as-is* and *system-to-be* models, respectively. A system-as-is model captures (i) the actors involved: primary users of the application as well as those indirectly affected by it, (ii) goals and plans of each actor, and (iii) dependencies among actors on their goals or plans. A system-to-be model introduces the solution as the system-to-be actor and its goals, plans, and dependencies. In the following phases, Tropos maps the system-to-be actor into one or more agents, and derives a detailed specification of agents' capabilities to implement on a chosen platform.

3. XIPHO

Xipho augments Tropos with tasks specific to CPA development in each development phase as shown in Table 1. Before describing Xipho's steps, we introduce a case study, and derive its system-as-is and system-to-be models.

Case Study: Ringer Manager

We demonstrate Xipho via a case study, which involves engineering the *Ringer Manager Application (RMA)*, a CPA for phones to help a user better handle incoming calls.

Automated ringer mode. The RMA sets an appropriate ringer mode on the user's phone based on the user's context at the time of an incoming call. The alternatives (assumed to be mutually exclusive) are to set the ringer mode to *silent*, *vibrate*, or *loud*.

Automated notification. If the user does not answer the call, the RMA sends a notification to the caller. The notification can be generic (e.g., "leave your name and number"), or describe the user's context at the time of missing the call, either abstractly (e.g., "in a meeting") or in detail (e.g., "in a meeting with Bob at the Starbucks").

We imagine the system-as-is as follows. Each episode starts when a *caller* tries to reach a *callee* by phone. The callee wants to be reachable unless he wants to work uninterrupted. The callee's plan is to answer the call if he wants to be reachable and not answer otherwise. The callee's decision to answer or not answer depends on (i) whether he disturbs a *neighbor* by answering or (ii) if the caller has a pressing need to reach him. In these cases, the callee depends on the neighbor or the caller to provide appropriate information. The callee sets an appropriate ringer mode on his phone to help him answer or not (e.g., loud to answer; silent to not). Next, when the callee does not answer a call, he would notify the caller of a reason (e.g., busy, in a class, talking to boss, and so on) or to ignore the call depending on who the caller is. If he decides to notify the caller, he would disclose only the necessary details, preserving privacy. Figure 1 shows the system-as-is model.

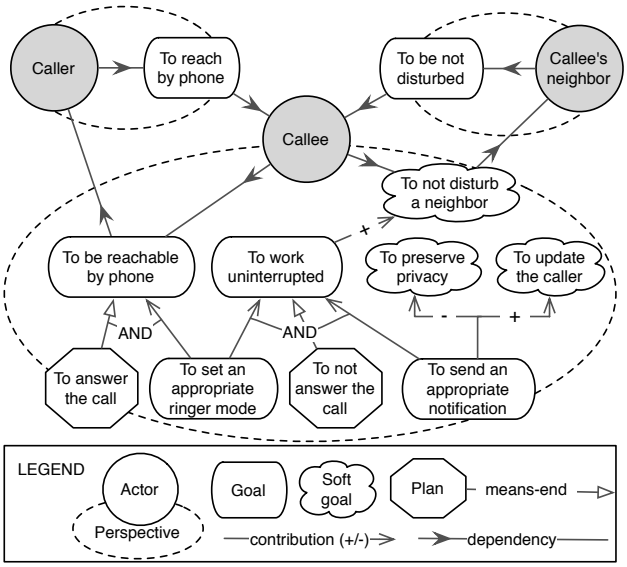

Figure 1: Actor model: Callee's perspective.

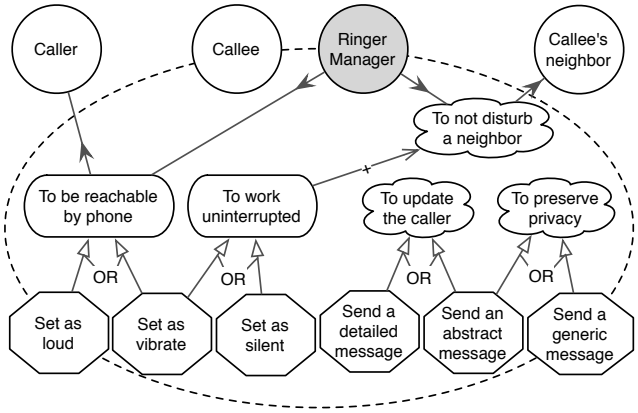

Figure 2: Expanding the RMA's perspective.

However, the system-as-is is quite inefficient. First, it relies on the callee to manually set an appropriate ringer mode and send an appropriate notification. Second, a caller has no effective way of expressing an urgent need to reach the callee—calling repeatedly does not help if the phone is silent. Third, it is tedious for a neighbor to let each callee know that the neighbor prefers not to be disturbed.

Figure 2 introduces the RMA (system-to-be) actor. The RMA acts on behalf of the callee and accordingly adopts the callee's goals to make call handling more efficient. This model captures what the RMA does and why, by linking the

Table 1: An overview of how Xipho extends Tropos to support CPA development.

Development Phase	Tropos Task	Xipho Task (Extension)
Early & late requirements	Identify system-as-is and system-to-be actors, and their goals, plans, and dependencies	Step 1: Identify contextual beliefs and resources
Architectural design	Define system's global architecture	Step 2: Derive a context information model
	Map system actors to software agents	Step 3: Incorporate a middleware agent
Detailed design	Specify agent capabilities and interactions	Step 4: Specify contextual capabilities
Implementation	Implement agents on a chosen platform	Step 5: Implement contextual capabilities

RMA's plans to the callee's goals (e.g., *set as silent* to accomplish the callee's goal *to be uninterrupted*). For simplicity, we explore the RMA only from the callee's (primary user's) perspective and assume that the secondary users (caller and neighbor) employ appropriate mechanisms to provide the information required by the RMA.

3.1 Step 1: Context-Means Analysis

A CPA needs to make decisions based on the contexts of one or more of the actors involved. The objective of *context-means analysis* is to identify scenarios where an actor's context provides the application a means of making a decision. We describe four types of such scenarios with examples.

OR-decomposition, where some decomposed goals or plans are to be chosen over the rest based on context, e.g., whether to set the ringer mode as *silent* or *vibrate*—vibrate may be acceptable in a seminar, but not in a one-on-one meeting.

Conflicting goals, where the goal chosen for accomplishment depends on context. For example, the RMA must choose between the callee's goals *to be reachable* and *to work uninterrupted*. This decision could be based on the callee's context—busy or not.

Soft goal, where the extent to which the goal is satisficed depends on context. For example, each ringer mode satisfices the soft goal *to not disturb a neighbor* differently. The chosen ringer mode could be based on who the neighbor is, e.g., set as *silent* near a colleague, *vibrate* near a family member, and *loud* near a stranger.

Dependency, where the dependum can be refined based on context. For example, the dependum *to be reachable* can be refined to context because the actual dependency is that the callee depends on the caller to provide contextual information (e.g., caller's context indicating whether he has a pressing need to reach the callee).

A developer must identify all scenarios of each type described above and perform one of the following for each.

- If the scenario is influenced by the primary user's (*callee*) context, capture the influence as a belief and add context-means links from the belief to each goal or plan involved in the scenario.
- If the scenario is influenced by a secondary user's (*caller* or *neighbor*) context, capture the context as a resource and add (or update) a dependency with the context resource as the dependum. Also, add context-means links from the resource to each goal or plan involved in the scenario.

The motivation is that the CPA would maintain beliefs about its primary user, and access information resources about the secondary users as shown in Figure 3.

3.2 Step 2: Context Information Modeling

We used a *context abstraction* as a placeholder for each contextual belief or resource in the previous step. Now, we systematically refine these abstractions in the CPA's scope for two reasons: (i) a motivation for model-driven development is to document the developer's thought process by explicitly capturing his or her intuitions, and (ii) a detailed model would yield a detailed specification, potentially making the implementation easier. To derive a CPA's context information model, identify:

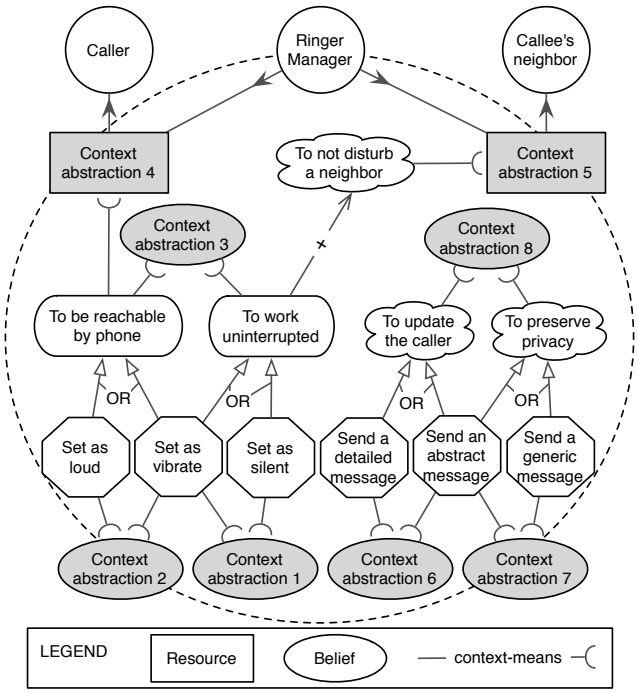

Figure 3: The RMA after context-means analysis.

A context model for each generic abstraction found in the previous step such that the context model consists of cognitive abstractions sufficient to make the decision specific to the context-based scenario influenced by the corresponding generic abstraction. For example, in Figure 3, *Context abstraction 2* is used to decide whether to set the ringer mode as *loud* or *vibrate*. This generic abstraction can be modeled as a spatial abstraction, *Ambience*, assuming that all that matters in setting the ringer mode as *loud* or *vibrate* is the callee's ambience. The decision to not set as *silent* must already have been made, using *Context abstraction 3*.

Context instances of a context abstraction, where appropriate, such that a context instance represents all situations, within the scope of the corresponding abstraction,

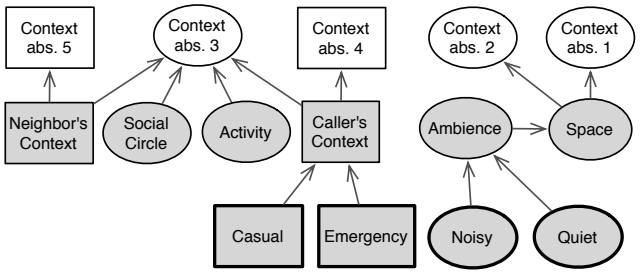

Figure 4: RMA's context information model.

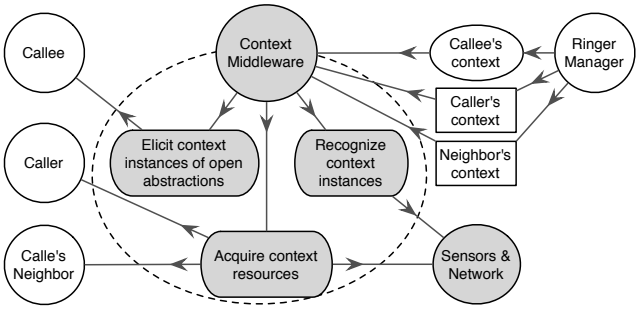

Figure 5: Introducing a middleware actor.

that lead to the same context-based decision. For example, context instances of *Ambience* could be *Noisy* and *Quiet* assuming that the ringer mode would be *vibrate* if the ambience is *Quiet* and *loud* if *Noisy*. However, to gain adaptivity, it would be desirable to elicit the instances of an abstraction at run time. We refer to such an abstraction as an *open abstraction*. For example, modeling the callee's *activity* as an open abstraction helps us elicit from the user which of his activities are to be uninterrupted.

Figure 4 shows a context information model derived for the RMA (each context instance is highlighted with a thick border). We restrict the case study to the ringer management aspect of the RMA (omitting notifications) for simplicity. We employed the context abstractions of *space*, *activity*, and *social circle* described in an existing metamodel [16], which however is loosely coupled with Xipho.

3.3 Step 3: Context Middleware

Once we derive a CPA-specific context information model, how can we enable the CPA to acquire this information? To acquire context information, the CPA must:

Elicit context instances of each open context abstraction from the primary user, e.g., elicit from the callee that the desired instances of the *Social circle* abstraction are *Family*, *Colleagues*, and *Others*.

Recognize context instances of each context abstraction from sensors, e.g., recognize the callee's *Ambience* as *Quiet* via data from the microphone on the callee's phone.

Acquire context resources from the secondary users, e.g., acquire from the caller that the *Caller's context* is *Casual*.

These are nontrivial tasks, but common to all CPAs. Although we model context information specific to each CPA, the abstractions can overlap as a user employs multiple CPAs (especially, since the abstractions are high-level, cognitive constructs). Thus, Xipho incorporates a middleware architecture in which (i) a reusable middleware agent elicits and recognizes a user's contexts, and (ii) multiple CPAs interact with the middleware to acquire desired contexts.

Figure 5 shows the middleware actor in the RMA's architecture. Importantly, such middleware components have been realized, e.g., [2, 14, 16]. Xipho can incorporate any middleware that achieves the goals specified in Figure 5.

3.4 Step 4: Contextual Capability Modeling

The objective of this step is to obtain a detailed agent specification to simplify the implementation. Xipho's specification of a CPA is a set of *contextual capabilities*, each conditioned on a context abstraction at an instance. A contextual capability can be represented as a rule, e.g., *Ambience = Noisy → Set as loud*. A developer performs the

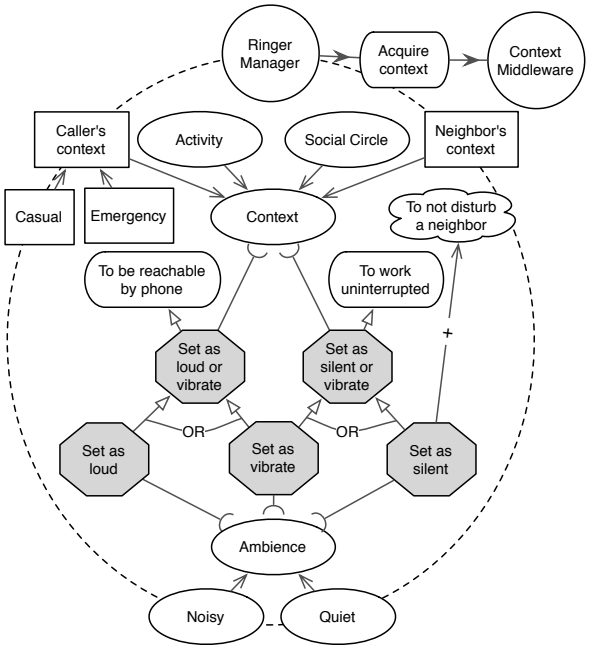

Figure 6: RMA's final model.

following substeps to derive an application specification.

- Identify agent capabilities, e.g., choosing and executing a plan, choosing a goal to accomplish, and managing a dependency.
- Identify the context abstraction that conditions each capability, replacing the generic context abstraction placeholders of Step 1 with specific abstractions from Step 2.

Figure 6 shows a refined actor model of the RMA that combines context abstractions and (highlighted) capabilities. This model can be used to generate a detailed set of contextual capabilities. Table 2 lists the contextual capabilities we identified for the RMA. We use variables ($?A_1$, $?S_1$, etc.) to capture instances of open abstractions.

3.5 Step 5: Implementation

Now, a developer must implement techniques to

Interact with the middleware to determine (i) elicited instances of open abstractions, e.g., to determine that instances of *Activity* are *Working*, *Driving*, and *Dining*, and (ii) the instance at which an abstraction is at a given time, e.g., to determine if *Activity = Working*, now.

Substitute variables in a contextual capability with in-

Table 2: RMA specification.

ID	Contextual Condition	→ Capability
C_1	*Activity* = $?A_1$ ∧ *Social circle* = $?S_1$ ∧ *Neighbor's context* = $?N_1$ ∧ *Caller's context* = *Emergency*	→ *Set as loud* ∨ *Set as vibrate*
C_2	*Activity* = $?A_2$ ∧ *Social circle* = $?S_2$ ∧ *Neighbor's context* = $?N_2$ ∧ *Caller's context* = *Casual*	→ *Set as silent* ∨ *Set as vibrate*
C_3	C_1 ∧ *Ambience* = *Quiet*	→ *Set as vibrate*
C_4	C_1 ∧ *Ambience* = *Noisy*	→ *Set as loud*
C_5	C_2 ∧ *Ambience* = *Quiet*	→ *Set as silent*
C_6	C_2 ∧ *Ambience* = *Noisy*	→ *Set as vibrate*

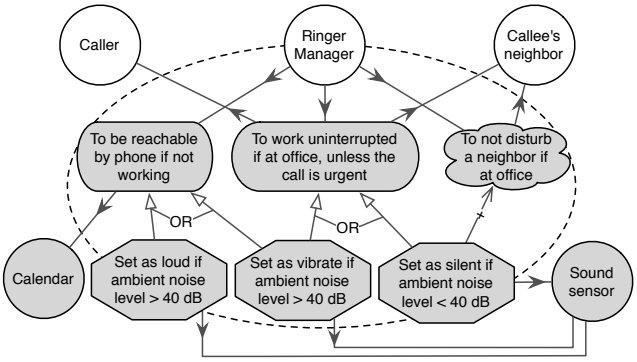

Figure 7: A model derived without Xipho.

stances elicited by the middleware, e.g., substitute *Activity* = $?A_1$ with *Activity* = *Driving* ∨ *Activity* = *Dining*.

Exercise capabilities by executing the rules derived in the previous section, e.g., exercise *Activity* = *Working* → *Set as silent* ∨ *Set as vibrate*.

4. COMPARISON

Xipho is an AOSE methodology tailored specifically for CPA development. However, what are the benefits of Xipho? To answer this, we compare the Xipho's RMA model shown in Figure 6 with an alternative model shown in Figure 7.

The alternative model enhances goals and plans in Figure 2 to incorporate context. The alternative model represents three major mistakes a developer, not applying Xipho, can commit. The alternative model:

Lacks explicit semantics of context, whereas Xipho models convey the semantics described in Steps 1 and 2. Such lack leads to ambiguity in interpreting a model. For example, Figure 7 is not clear about whether two goals (e.g., *... if not working...* and *... if at office...*) conflict; similarly, the figure does not convey that the ringer mode must be *silent* only if the call is not urgent and the callee's ambience is quiet.

Introduces redundancy in the context information model, whereas Xipho removes it by incorporating cognitive constructs. For example, each goal in Figure 7 describes a contextual condition involving activities, redundantly. In contrast, Figure 6 simplifies the model by including the *Activity* open abstraction.

Does not separate the concerns of context acquisition from the RMA design, whereas Xipho cleanly separates the RMA's and middleware's concerns. For example, Figure 7 describes noise levels used to decide a ringer mode within RMA's design. In contrast, Figure 6 delegates such tasks to the middleware, simplifying the RMA's design.

5. EMPIRICAL EVALUATION

The foregoing intuitions lead us to hypothesize that Xipho (i) reduces the time and effort required to develop a CPA, and (ii) enhances the comprehensibility of CPA designs (for other developers). We evaluated our hypotheses in a developer study, comparing Xipho against the baseline of Tropos.

5.1 Study Design

Our subjects were 46 students of a graduate level computer science course. Each subject worked in a team of three (12 teams), two (three teams), or one (four subjects) of his or her choosing. Subjects received a partial grade toward course credit for producing all deliverables. The study was approved by the Institutional Review Board (IRB) and we obtained an informed consent from each subject. Nonparticipants could work on an alternative task.

We conducted the developer study in three phases.

Practice, which required each team to learn one of Tropos or Xipho, and exercise it to model an application, which we reviewed to help them understand.

Modeling, which required each team to model a second application. No feedback was given during this phase.

Verification, which required each subject (working solo) to verify an existing model of a third application for completeness and comprehensibility.

The study lasted for eight weeks—two weeks per phase for subjects and a week each between phases for us to review.

5.1.1 Study Units

Subjects completed a prestudy survey about their experience in software modeling and development (of context-aware and mobile applications, and in general). Using this information, we formed three groups of teams such that teams' skill sets and sizes balanced across groups.

In the first phase, we assigned one of the following applications to each group. In the subsequent phases, we rotated the assignments as shown in Table 3. For each application, we provided the following description and three–four scenarios in which the application must employ the (primary or secondary) user's context.

Smart alarm, whose objective is to advance or postpone an alarm on a user's phone. The application must employ context information such as (i) traffic and weather conditions, (ii) the amount of rest the user has had by the time alarm goes off, and (iii) changes to the user's schedule of upcoming events.

Lifestyle motivator, which helps a user find friends and identify physical activities to perform with those friends. The application must employ context information such as (i) the proximity to appropriate facilities and desired friends, (ii) the weather conditions, and (iii) the amount of exercise the user has already had.

Intelligent reminder, which decides when and how to remind a user of events. The application must employ context information such as (i) convenient resources (e.g., remind on the computer instead of the phone), (ii) importance (e.g., reply to the boss), and (iii) timeliness (e.g., pick up a book when near the library).

Table 3: Application assignments in each phase.

Phase	Task	Group 1	Group 2	Group 3
1	Practice	Alarm	Motivator	Reminder
2	Modeling	Motivator	Reminder	Alarm
3	Verification	Reminder	Alarm	Motivator

5.1.2 Alternatives

Teams within each group (1, 2, or 3) were divided into two groups (again, with balanced skill sets and team sizes).

Control group (C): provided with the modeling primitives, a description, and example models from Tropos.

Xipho group (X): provided with the steps, modeling primitives, and the RMA example.

5.1.3 Deliverables

In each of the first and second phases, subjects were given an application specification (and a methodology) and asked to deliver (i) a model of the application (covering all specified scenarios); (ii) responses to a survey after each work-session recording the time spent (in hours and minutes), and effort expended (on a scale of 1∼*easy* to 7∼*difficult*) during the session. The survey also required subjects to record a detailed breakdown of tasks they performed in each session.

In the third phase, each subject was given the specification of a third application and two models of it produced by other subjects (one from Control and one from Xipho team) in the second phase. A description of the Xipho's primitives was provided to those who did not apply Xipho in earlier phases. Our motivation was to compare the comprehensibility of models by subjects experienced with Xipho and those new to it. The subjects were asked to verify the model and answer a survey to record (i) a rating of how complete the model was (on a scale of 1∼*incomplete* to 7∼*complete*), and (ii) a rating of how comprehensible a model was (on a scale of 1∼*easy* to 7∼*difficult*) relative to each context-based application scenario the model handled.

We allowed teamwork in the first two phases, but required solo work in the third because (i) such ratings are inherently subjective, and (ii) the third phase was less demanding than the first two, and we expected subjects to complete the deliverables (in time) without compromising quality.

5.2 Results

We performed two-tailed (i) t-test to compare the difference in mean (μ) time spent, (ii) F-test to compare the difference in the variances (σ^2) of time spent, and (iii) Wilcoxon's *ranksum*-test to compare the difference in the median ($\widetilde{x}$) of effort ratings during Practice and Modeling [12]. We compared medians for the effort ratings since rating is ordinal.

With respect to the Practice and Modeling phases, Figure 8 shows box plots of times spent, and Figure 9 shows effort perceived, building models (there were no instances of the lowest rating). The figures also show the result of hypothesis testing (* and ** indicate sufficient evidence to reject the corresponding null hypothesis at significance levels of 10% and 5%, respectively).

Recall that each model produced in Modeling was rated by more than one subject for completeness and comprehensibility during Verification. First, we measured the interrater reliability for ratings of each model using Krippendorff's alpha (α) [13] ($\alpha = 1$ indicates perfect reliability and $\alpha = 0$, the absence of reliability; $\alpha = 0.67$ is a suggested lower bound for filtering). We excluded all models with $\alpha < 0.67$ from further analysis. Next, we performed *ranksum*-tests to test if the ratings differed significantly. Figure 10 shows a comparison, and the results of hypothesis testing.

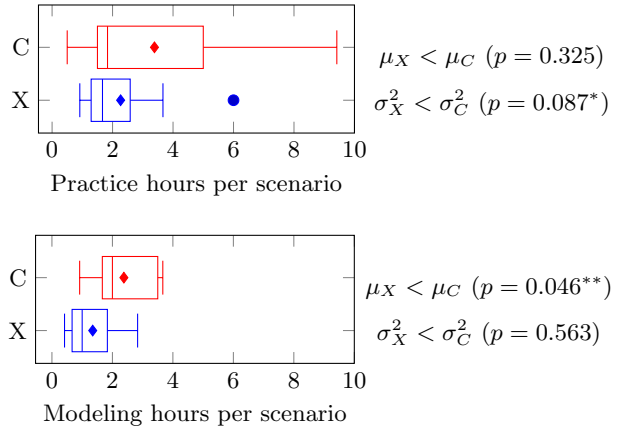

Figure 8: Time spent building models.

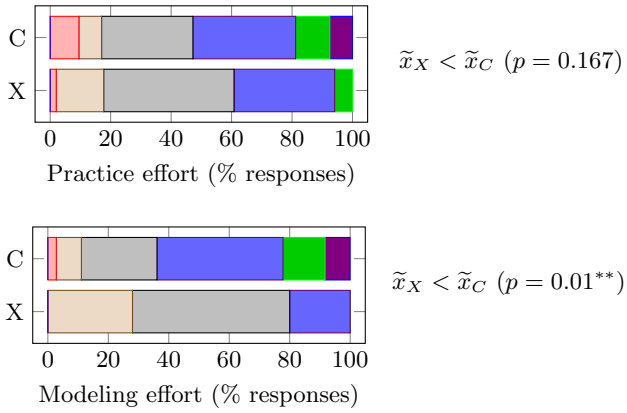

Figure 9: Perceived effort in modeling.

5.3 Discussion

5.3.1 Time and Effort of Model Building

We found that the time and effort expended by a Xipho team in the Modeling phase are significantly lower than those expended by a Control team ($p = 0.046$ and $p = 0.01$, respectively). This validates that Xipho, a methodology tailored for CPA development, can simplify the development process as opposed to a generic methodology.

Next, we found that the differences in time spent and effort were not significantly different during the Practice phase (when teams were learning Xipho or Tropos). However, we made an important observation—the variance in time spent was significantly larger for Control than Xipho teams, although Control teams had fewer primitives to learn than Xipho teams. We speculate that such difficulties could lead a developer, especially a beginner, to give up modeling. Thus, a methodology with systematic steps specific to CPAs is essential for the success of model-driven development of CPAs.

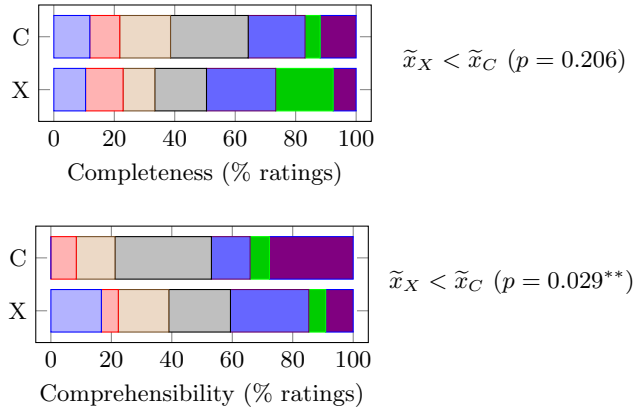

Figure 10: Subjective ratings for completeness and comprehensibility.

5.3.2 *Completeness and Comprehensibility*

We found that the completeness ratings of Xipho and Control models did not differ significantly. The result was not surprising, though, for two reasons. (1) Conceptually, the modeling primitives of Tropos are sufficient to model any of the assigned applications. Thus, neither of the groups has an advantage from this perspective. (2) Each team, irrespective of the group, had an incentive (project grade) to produce complete models (even if it took more time).

Next, we found that comprehending a Xipho model could be significantly easier than a Control model. This is an important result since Xipho's ability to yield comprehensible models is a major benefit of employing it. To understand why this is the case, we compared the modeling constructs used by Control and Xipho teams, as shown in Table 4. We found that compared to Control, Xipho models employed:

Shorter textual descriptions within primitives. We attribute a Xipho model's conciseness to employing the explicit context modeling semantics provided by Xipho. Further, our results suggest that concise Xipho models are easier to comprehend than Control models' subjective and verbose textual descriptions.

Fewer modeling primitives indicating that Xipho models reused contextual beliefs and resources across goals and plans. We conjecture that employing fewer primitives helps reduce information overload and leads to models that are easier to comprehend.

Fewer actors and dependencies indicating that Xipho models delegated the concerns of context acquisition to the middleware, whereas Control models employed additional actors (with associated, goals and plans) to acquire context. Also, a Control developer must deal with context acquisition for each application, whereas a Xipho developer needs to understand the middleware just once.

6. RELATED WORK

Xipho begins with cognitive notions and systematically leads to an opportunity to exploit reusable components. Xipho helps bridge AOSE and context-aware systems.

Context-Aware Systems

A wealth of context-aware systems research [2] focuses on providing architectural support for developing context-aware applications, usually via a middleware, e.g., [11, 14, 16, 19], to protect developers from low-level concerns of context acquisition. Often, such architectures provide a generic context metamodel [4], which an application developer must tailor during development. Sollenberger and Singh [21] show that developers employing a systematic methodology benefit more from a middleware than those who do not. CPA developers can employ Xipho with any middleware that can elicit and reason about context instances. Bolchini et al. [5] describe a context-aware system that centrally manages contextual data. In contrast, Xipho's middleware agent manages a user's contextual data locally and advocates users to interact for satisfying dependencies on contextual resources.

Table 4: Modeling primitives employed.

Primitive	Count		p-value	Text len.		p-value
	C	**X**	$\tilde{\mathbf{x}}_\mathbf{X} < \tilde{\mathbf{x}}_\mathbf{C}$	**C**	**X**	$\tilde{\mathbf{x}}_\mathbf{X} < \tilde{\mathbf{x}}_\mathbf{C}$
Actor	6.0	3.4	0.02^{**}	77	33	0.02^{**}
Dependency	8.2	4.2	0.02^{**}	–	–	–
Goal	4.4	2.6	0.57	136	58	0.13
Soft goal	5.8	0.8	0.01^{**}	231	21	0.01^{**}
Plan	9.8	7.4	0.29	365	145	0.06^{*}
Belief	0.0	3.6	–	0	45	–
Resource	0.0	2.0	–	0	30	–
Overall	*34.2*	*24*	*0.09**	*809*	*332*	*0.02***

AOSE Methodologies

AOSE promotes development activities to capture high-level abstractions, e.g., agents and goals over low-level abstractions, e.g., classes and methods. Xipho extends Tropos to explicitly deal with the cognitive notion of context.

Ali et al. [1] capture context-based scenarios as variation points and employ them to analyze requirements. Whereas their focus is to analyze a contextual goal model to detect conflicts and inconsistencies among requirements, Xipho deals with context in requirements as well as in design and implementation. Zacarias et al. [23] describe a context-aware agent-oriented ontology for modeling human agents in an enterprise setting. Whereas they provide a metamodel (ontology) to describe the dynamic and situated human behavior, Xipho provides systematic steps to exploit such a metamodel for developing CPAs.

Xipho's context analysis, modeling, and specification steps are generic, and can be employed in other AOSE methodologies. Prometheus [22] can be extended to incorporate context as percepts and external data used to describe the environment. Xipho can be incorporated into ADELFE [3] to characterize the environment, where ADELFE's active and passive entities correspond to Xipho's context instances and open abstractions. Rahwan et al. [17] describe a social modeling extension to the ROADMAP methodology. Xipho brings together a social model, users' activities, and spatial attributes into a context model, which can be linked a role's responsibilities, generalizing Rahwan et al.'s extension.

Other Methodologies

Ceri et al. [7] employ WebML to model context and context-triggered adaptive actions in a user-independent manner. Serral et al. [18] employ a domain-specific PervML to model context followed by auto generation of code that represents and handles context. Henricksen and Indulska [10] employ

CML (Context Modeling Language) to model context and preference modeling to rank context-based choices of a user. These methodologies describe how to model context, assuming that a developer knows what aspects of context to model and why. Xipho helps a developer not only to model context, but also to arrive at meaningful context abstractions and instances specific to an application scenario.

7. CONCLUSIONS AND DIRECTIONS

This paper has shown how an agent-oriented methodology can be enhanced to engineer CPAs, an important problem that has not been adequately addressed in the literature. Our findings about the benefits of Xipho bear important implications for its practical adoption. A huge market for CPAs already exists in the form of smart phone applications (such as RMA). Further, we conjecture that applications produced using Xipho offer a more natural user experience than those built conventionally. We are considering ways to design a tractable study to evaluate this claim.

Xipho provides a key component of our overall vision of developing CPAs. On the one hand, Xipho extends the benefits of a goal-based methodology (specifically, Tropos) to CPAs—context is arguably naturally understood via cognitive notions. On the other hand, Xipho builds a CPA on a reusable middleware, separating the concerns of context acquisition from CPA development. Our implementation of the middleware elicits a user's context instances and learns to recognize them from sporadic sensor data via semi-supervised machine learning [9]. Our ongoing work seeks to reduce user effort in context elicitation via active learning.

Systematically contextualizing cognitive notions such as privacy and trust is largely unexplored. A contextual approach to privacy and trust could facilitate applications with multiple interacting CPAs. Such an extension could build on existing approaches for finding social relationships between agents sharing a context [15], and coordinating agents by transparently propagating the context information [14].

Acknowledgments

We thank the National Science Foundation for support under grant 0910868, and Fatma Başak Aydemir, John Mylopoulos, and the anonymous reviewers for helpful comments.

8. REFERENCES

[1] R. Ali, F. Dalpiaz, and P. Giorgini. Reasoning with contextual requirements: Detecting inconsistency and conflicts. *Inf. Softw. Technol.*, 55(1):35–57, Jan. 2013.

[2] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.

[3] C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard. Engineering adaptive multi-agent systems: The ADELFE methodology. In B. Henderson-Sellers and P. Giorgini, eds., *Agent-Oriented Methodologies*, ch. 7, pp. 107–135. Idea Group, Hershey, PA, 2005.

[4] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Perv. Mob. Comput.*, 6(2):161–180, Apr. 2010.

[5] C. Bolchini, G. Orsi, E. Quintarelli, F. A. Schreiber, and L. Tanca. Context modeling and context awareness: Steps forward in the Context-ADDICT project. *IEEE Data Eng. Bull.*, 34(2):47–54, 2011.

[6] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *JAAMAS*, 8(3):203–236, May 2004.

[7] S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven development of context-aware web applications. *ACM TOIT*, 7(1), Feb. 2007.

[8] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, Dec. 2001.

[9] C.-W. Hang, P. K. Murukannaiah, and M. P. Singh. Platys: User-centric place recognition. In *AAAI Workshop on Activity Context-Aware Systems*, 2013.

[10] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: Models and approach. *Perv. Mob. Comput.*, 2(1):37–64, Feb. 2006.

[11] J. Hong, E.-H. Suh, J. Kim, and S. Kim. Context-aware system for proactive personalized service based on context history. *Expert Syst. Appl.*, 36(4):7448–7457, May 2009.

[12] N. Juristo and A. M. Moreno. *Basics of Software Engineering Experimentation*. Kluwer, 2001.

[13] K. Krippendorff. Reliability in content analysis. *Human Comm. Res.*, 30(3):411–433, 2004.

[14] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications: The TOTA approach. *ACM TOSEM*, 18(4):15:1–15:56, July 2009.

[15] P. K. Murukannaiah and M. P. Singh. Platys Social: Relating shared places and private social circles. *IEEE Internet Computing*, 16(3):53–59, May 2012.

[16] P. K. Murukannaiah and M. P. Singh. Platys: An empirically evaluated middleware for place-aware application development. TR 2014-2, North Carolina State University, Feb. 2014.

[17] I. Rahwan, T. Juan, and L. Sterling. Integrating social modelling and agent interaction through goal-oriented analysis. *Comput. Syst. Sci. Eng.*, 21(2), 2006.

[18] E. Serral, P. Valderas, and V. Pelechano. Towards the model driven development of context-aware pervasive systems. *Perv. Mob. Comput.*, 6(2):254–280, 2010.

[19] Q. Z. Sheng, S. Pohlenz, J. Yu, H. S. Wong, A. H. H. Ngu, and Z. Maamar. ContextServ: A platform for rapid and flexible development of context-aware web services. In *Proc. ICSE*, pp. 619–622, 2009.

[20] M. P. Singh. Self-renewing applications. *IEEE Internet Computing*, 15(4):3–5, July 2011.

[21] D. J. Sollenberger and M. P. Singh. Kokomo: An empirically evaluated methodology for affective applications. In *Proc. AAMAS*, pages 293–300. 2011.

[22] M. Winikoff and L. Padgham. *Developing Intelligent Agent Systems: A Practical Guide*. Wiley, Chichester, UK, 2004.

[23] M. Zacarias, H. S. Pinto, R. Magalhães, and J. Tribolet. A 'context-aware' and agent-centric perspective for the alignment between individuals and organizations. *Info. Syst.*, 35(4):441–466, June 2010.