2012

Improved initial class diagrams with ATSA:OO

Robert B. K Brown
University of Wollongong, bobbrown@uow.edu.au

Angela Piper
University of Wollongong, apiper@uow.edu.au

Publication Details

Brown, R. B. K. & Piper, A. M. E. (2012). Improved initial class diagrams with ATSA:OO. 21 st International Conference on Information Systems Development (ISD 2012) (pp. 1-12). Spinger.

Improved initial class diagrams with ATSA:OO

Abstract
Despite the popularity of Object Oriented programming and design, their analysis phase is difficult to learn and relies heavily on the analyst's intuition and experience. Even industry practitioners can produce initial class diagrams requiring time-consuming refinements in the design phase. Business clients can find it difficult to follow its diverse techniques and constructs. The A TSA method offers analysis under a single theoretical framework that is specifically designed to be easily understood by both neophyte practitioners and business clients. However as it follows the procedural paradigm, it is not suitable for direct 00 application. This paper presents an adaptation of A TSA for the 00 paradigm, which produces an initial class diagram, complete with suggested attributes and methods. It offers improved cohesion and comprehensibility over its 00 counterpart.

Keywords
class, diagrams, atsa, oo, improved, initial

Disciplines
Engineering | Science and Technology Studies

Publication Details
Brown, R. B. K. & Piper, A. M. E. (2012). Improved initial class diagrams with ATSA:OO. 21 st International Conference on Information Systems Development (ISD 2012) (pp. 1-12). Spinger.

Improved Initial Class Diagrams with ATSA:OO

Robert B. K. Brown and Angela M. Piper

Abstract Despite the popularity of Object Oriented programming and design, their analysis phase is difficult to learn and relies heavily on the analyst's intuition and experience. Even industry practitioners can produce initial class diagrams requiring time-consuming refinements in the design phase. Business clients can find it difficult to follow its diverse techniques and constructs. The ATSA method offers analysis under a single theoretical framework that is specifically designed to be easily understood by both neophyte practitioners and business clients. However as it follows the procedural paradigm, it is not suitable for direct OO application. This paper presents an adaptation of ATSA for the OO paradigm, which produces an initial class diagram, complete with suggested attributes and methods. It offers improved cohesion and comprehensibility over its OO counterpart.

1 Introduction

A previous paper (Brown and Piper 2011) briefly presented the Activity Theoretic Systems Architecture (ATSA) methodology. Activity Theory (AT) is use-centric; based upon a hierarchy of doings that occur within any given human endeavor. This makes it more compatible with the mindset of business clients (who are more easily able to describe their domain in terms of tasks performed and *goal*s satisfied) and, furthermore, makes the resulting system less susceptible to changes (such as those arising from staffing reconfigurations).

ATSA yielded system specifications in the form of a data dictionary with transforms noted in pseudocode fragments, anticipating a procedural implementation. However, the majority of the steps within ATSA were agnostic of implementation paradigm. Thus it was anticipated that ATSA could be adapted to suit the widely used Object Oriented (OO) paradigm.

Robert B.K. Brown
University of Wollongong, 2522 NSW AUSTRALIA, bobbrown@uow.edu.au

Angela M. Piper
University of Wollongong, 2522 NSW AUSTRALIA, apiper@uow.edu.au

Object orientation provides encapsulation, polymorphism and inheritance which facilitate maintainability, reuse and modification/extension. These advantages have made OO approach immensely popular.

Of the three phases of the OO approach — Object Oriented Analysis (OOA), Object Oriented Design (OOD) and Object Oriented Programming (OOP) — the majority of these advantages are found in the latter two phases, while the first phase remains problematic. Despite the notion that OOA is "based on concepts that we first learned in kindergarten." (Coad and Yourdon 1991), it is notoriously difficult for all but the most experienced practitioners to apply (Steven et al. 1974, Holmboe 2004, Svetinovic et al. 2005).

As ATSA was designed with learnability in mind (Brown 2010), an OO version of ATSA should be significantly easier for inexperienced practitioners to apply. Therefore Object Oriented ATSA (ATSA:OO) should provide an excellent pathway from early phase elicitation into OOD and OOP.

This paper briefly recaps the ATSA method; presents new class finding heuristics underlying ATSA:OO and provides a brief worked example based on the coffee machine problem drawn from a typical OO teaching text. The example demonstrates that ATSA:OO can generate a workable initial design class diagram.

2 Initial Design: Model-0

In one of the earliest descriptions of the OO concept (Stevens et al. 1974) two phases of design were described. An initial *abstract* design, known as a General Design, emerged from an architectural system-level consideration of the problem description, which would then be passed on for Detailed Design. This description effectively created OOA.

Most current OO approaches and examples present an initial design, arising from (often disappointingly cursory) OOA considerations, typically employing variations of the Use Case diagram technique. There are many (often subtle) variations of this OOA, using different terms for their resulting design. We adopt the general term model-0 to refer to the outcome of any OOA method. In each case, model-0 serves as the basis for ongoing iterative design revision.

Our intention is for ATSA:OO to sit in the place of OOA, offering Activity Theoretic analysis techniques in place of the many Use Case techniques, and therefore we expect ATSA:OO to generate a model-0 which is at least as good a starting point for further design refinement as that generated by OOA.

We will detail the generation of a model-0 from a typical textbook problem, and compare it to a corresponding model-0 from a typical OOAD approach, independantly conducted by a practitioner from the software industry.

3 The Coffee Machine Problem

To illustrate ATSA:OO, we will address a problem drawn from a typical OO teaching text (Martin 1995 p60). The problem describes a simple drip-feed coffee brewing machine with its primary physical components and operations described as follows:

> "The Mark IV special makes up to 12 cups of coffee at a time. The user places a filter in the filter holder, fills the filter with coffee grounds, and slides the filter holder into its receptacle. The user then pours up to 12 cups of water into the water strainer and presses the "Brew" button. The water is heated until boiling. The pressure of the evolving steam forces the water to be sprayed over the coffee grounds, and coffee drips through the filter into the pot. The pot is kept warm for extended periods by a warmer plate, which only turns on if there is coffee in the pot. If the pot is removed from the warmer plate while coffee is sprayed over the grounds, the flow of water is stopped, so that brewed coffee does not spill on the warmer plate. The following hardware needs to be monitored or controlled.
>
> - The heating element for the boiler. It can be turned on or off.
> - The heating element for the warmer plate. It can be turned on or off.
> - The sensor for the warmer plate. It has three states: warmerEmpty, potEmpty, and potNotEmpty.
> - A sensor for the boiler, which determines if there is water present or not. It has two states: boilerEmpty or boilerNotEmtpy.
> - The brew button. This momentary button starts the brewing cycle. It has an indicator that lights up when the brewing cycle is over and the coffee is ready.
> - A pressure-relief valve that opens to reduce the pressure in the boiler. The drop in pressure stops the flow of water to the filter. It can be opened or closed."

We briefly report on an analysis of this problem using ATSA:OO, treating the problem description text (above) as the information elicited from a client. The initial two phases of the method run as previously described for the procedural ATSA. For brevity we omit that elicitation and the construction steps of the activity network, but offer a brief recap of the ATSA method as previously presented.

4 Activity Theoretic Systems Architecture (ATSA)

ATSA employs a single coherent theory throughout the Systems Analysis and Design (SA&D) process, Activity Theory (AT). It is well beyond the scope of this paper to properly introduce AT, so the interested reader is invited to consult earlier publications (Vygotsky 1978, Leont'ev 1978, Engström 1987)

The construction of ATSA was motivated by a concern that neophyte practitioners lacked the experience and tacit skills required under other methods and approaches, and a desire to address the persistently poor success rates reported for IT projects (Crear 2009).

To establish the basis of the extension offered in this paper, a brief recap of the three main phases of the ATSA method is presented below. A more complete description of the method can be found in Brown and Piper I. (2011).

4.1 ATSA Elicitation

The analyst identifies a list of *roles*, each being some human (who typically occupies some defined '*position'*) acting in a specific mindset, engaged in task-specific work and, typically using task-specific tools. Any one position may fill several roles, but there may also be more than one position capable of, or authorized to, act in a given role.

As the final use-centric system will be described to facilitate doings, ATSA considers *its* actors to be roles not positions. ATSA designs for a *community* of such actors (being roles performing activities) and designs both an overall facilitating system and appropriate *activity*-specific tools for each. Positions therefore, cease to be of immediate concern to the analyst.

The analyst identifies Candidate Instruments (CIs) from existing files, folders, forms, records, registers, lists, databases. These are data-like *things* (tangible or otherwise) which are required for roles to perform their work and satisfy their goals, or which are made available for other roles as a result of their work.

4.2 ATSA Analysis

Having obtained the obvious business details, the analyst identifies task-specific goals. These goals are more readily obtained than the more abstract activity-level motives (to disambiguate an OO object from an AT object, we will use the term motive for the latter). When conflated with the single roles that work to satisfy each goal, and also the list of all CIs both required and produced in the course of that work, these are labeled Goal Driven Actions (GDAs).

As they will connect together in transactional chains, GDAs may be envisaged as the nodes of a directed graph, with CI transactions as arcs. For convenience, this may be rendered as a GDA adjacency matrix (GDAAM).

Goals, however, only exist at the action layer of AT. Activities are driven by more abstract and somewhat strategic motives which ATSA envisaged as *coherent sets of consistent goals*.

To assist in identify activities, GDAs may be broken into component pieces and conflated following AT precepts. The analyst decomposes each GDA into a number of Single Instrument Nodes (SINs) equal to the number of CIs it receives and outputs. Each SIN inherits its parent GDA's goal and constraints. AT requires that each activity have just one role as the doer and a single motive (coherent set of

consistent goals), so SINs are sorted first by role, and then into groupings of consistent goals. Some subjective judgment on the part of the analyst is indicated here, but it is informed by an understanding garnered during elicitation, and is at, worst, no broader an application of experience than that called for in any other method or approach (experience suggests it will be less). Under ATSA, the clients may still actively participate and assist as required, as all ATSA models, including the final activity network, are expressed in terms comprehensible to them.

Detailed piecewise deconstruction of GDAs into SINs goes some way toward revealing duplications, inefficiencies and ambiguities; however it has been found that with experience, clustering of GDAs to Activities directly may be possible. The SIN sorting technique might in fact be used topically, as a check. For learners, SIN sorting serves as 'training wheels'.

A growing familiarity with their client's business processes can tempt analysts to institute changes prematurely. Instead, they must refrain, and record any such 'good ideas' for consideration in the next phase.

Activities identified and conflated from the coherent grouping of GDAs (or their SIN fragments) will each have numerous CIs, both in and out, often connecting them together. The resultant activity network may be illustrated as a directed graph and/or represented as an adjacency matrix known as the Combined Activity Table (CAT).

4.3 ATSA ReDesign

In consultation with the stakeholder as needed, the analyst seeks to reconfigure and rearrange the client's processes to enhance efficiency and to relieve the roles of burdensome lower-order doings (which have simple drivers such as conditions or simple goals).

Any 'good ideas' noted during preceding phases may now be considered as options for changes. Further suggestions for rationalization of the network can be found using Node Reduction Heuristics (NRHs) which were described in detail elsewhere (Brown et al. 2006).

The complete activity network from the analysis phase will highlight the interactional consequences of any changes and thus prompt more complete and carefully considered decisions.

In its procedural mode, ATSA then envisages a central system which mediates *automatable* activities, holds instrument values and conducts transforms upon them as required. The specification produced takes the form of a data dictionary and a series of transformation descriptions with constraints and conditions recorded (wherever possible) in structured English (pseudocode).

5 ATSA:OO

ATSA:OO follows the first two phases of ATSA exactly as described above. It also encourages redesign and refinement in the same way as ATSA, drawing upon 'good ideas' and the NRHs. To conform to the OO paradigm, ATSA:OO does not collapse automatable activities into an amorphous assortment of data-instruments and transforms. ATSA:OO retains all redesigned activities and transactions in their networked form, which remains comprehensible to the client. It then seeks to identify coherent classes and their associations from this modified network.

Table 1 shows the new elements of ATSA:OO in comparative sequence.

ATSA	ATSA:OO
Positions, Roles, CIs	
Goal Identification GDAs SINs Conflation of Activities Initial Activity Network	
NRHs, Good Ideas Client Input, Refinement New Activity Network, CAT	
Automate Activities Bend Instrument Paths to System	Apply Class Finding Heuristics
Convert Temporal Rules to Deontic Data Dictionary	Identify Classes Attributes and Methods
Pseudocode Transforms	Model-0 Class Diagram

Table 1. Elements of ATSA compared to Additional and Replacement ATSA:OO Elements

Since an activity represents a coherent unit of work, conducted by a role, to satisfy a coherent set of consistent goals, it is a good candidate to be the basis of a class. OO requires that a class has both methods (work done) and attributes (essentially a list of its variables), so we require that any class identified from within an activity network must consist of at least one activity and at least one instrument.

Classes will tend to be more complex than simple pairings of singleton activities and instruments. Figure 1 below, shows the complete activity network for the coffee maker problem. It also indicates identified classes by using grey zones encasing each of them. We offer four rules to guide the clustering of activities and instruments into classes. These are described below.

5.1 New Class Finding Heuristics

Since ATSA has already identified data-like and function-like constructs, namely CIs and activities, the primary problem with identifying classes from the activity

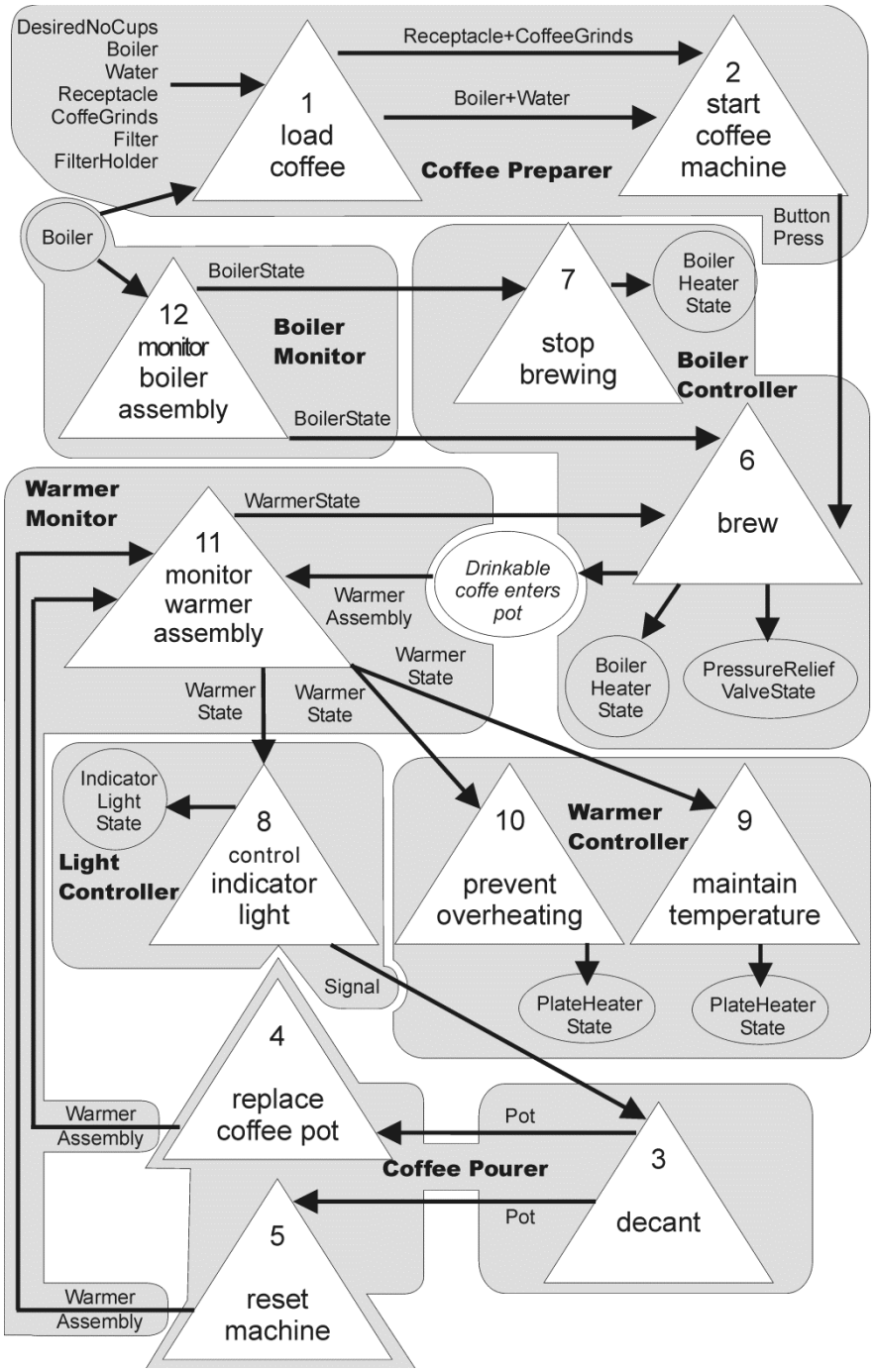

Fig. 1 Activity Network for the Coffee Maker with Classes identified.

network, is determining which instruments should be clustered with which activities. To address this problem we devised the four heuristics.

At each application of these heuristics, we first attempt to form one or more clusters (proto-classes) under the principle of that heuristic. Where this is not possible (for want of at least one available activity and available instrument) we extend one or more of the existing proto-classes. Priority is given to proto-classes suggested by preceding heuristics, over those suggested by later heuristics. ie: later heuristics cannot de-cluster proto-classes formed under preceding heuristics.

Rule 1. Cluster 'source' activities with their output instruments.

If an activity is responsible for outputting a given instrument to multiple other activities, we cluster the activity with that instrument. When many entities obtain data from a single source, good design practice says that the data should reside with the single source rather than with the many recipients.

In figure 1, activity 11 is a source activity which outputs multiple copies of the WarmerState instrument to different activities. Rule 1 therefore clusters activity 11 with the WarmerState instrument. This is sufficient to form a proto-class.

Similarly, activity 12 is clustered with the BoilerState instrument and activity 3 with the Pot instrument; each forming their own proto-class.

Rule 2. Cluster 'sink' activities with their input instrument(s).

If an activity is receives copies of an instrument from multiple other activities, we cluster the receiving activity with that instrument. This rule recognizes that rather than having the data reside one or other of these multiple sources, a single copy of the data should reside with the recent entity, which should permit those source entities to set the value of that data.

In figure 1, activity 11 is the recipient of the Warmer Assembly instrument from multiple activities. Rule 2 would therefore cluster activity 11 with the WarmerAssembly instrument, however activity 11 has already been clustered into a proto-class by an application of Rule 1 and is thus not available. The instrument WarmerAssembly is insufficient to form a proto-class on its own (lacking an activity) and thus it is clustered with the existing proto-class (which already contains activity 11 and the WarmerState instrument).

Rule 3. Cluster activities that input the same or similar instruments and that also output the same or similar instruments, with those instruments.

This Rule seeks to cluster according to a commonality of process (based on both data and function). Two or more entities which act upon variations of the same input data, producing variations of the same output data, are variations of essentially the same function type; and therefore should be clustered together with their input and output instruments.

In figure 1, activities 6 and 7 both receive variations of the BoilerState and produce various changes to BoilerHeaterState. BoilerState has already been clustered, however BoilerHeaterState is still available. Thus, it is still possible form a proto-class consisting of activities 6 and 7 and BoilerHeaterState.

Similarly, activities 9 and 10 are be clustered with PlateHeaterState, though WarmerState was unavailable.

Activities 4 and 5, however, have no available instruments for clustering as Pot and WarmerAssembly have each been clustered under the preceding heuristics. They are thus unable to form a proto-class by themselves, and must therefore be assigned to either the proto-class containing activity 11 and WarmerAssembly, or the one containing activity 3 and Pot. In this situation, were guided by roles identified in the ATSA analysis phase. Activities 4 and 5 are conducted by the same role as activity 3, so we decide to cluster them together.

Rule 4. Consider unclustered activities and their associated instruments.

After the application of the preceding three heuristics, a number of isolated activities and or instruments may remain unclustered. For these, we rely on the roles and motives identified under ATSA and the intuition of the analyst.

In figure 1, activities 1 and 2 are clustered with their outstanding input and output instruments due to a commonality of role and motivation identified under ATSA.

The only remaining unclustered entities are activity 8 and the IndicatorLightState and Signal instruments. These cluster into a proto-class. As an extremely simple class this may later be absorbed into another class, but this consideration would arise under OOD.

5.2 Identify Classes, Attributes and Methods

At the completion of the class finding heuristics, we have a number of proto-classes which can now be converted into classes. Class names are drawn from the roles and goals of the component activities, in the form of a new role name. For example: under initial analysis, activity 11's goal was to monitor the state of the WarmerAssembly, performed by a PressureSensor role. The class containing activity 11 and the WarmerAssembly and WarmerState instruments is given the name: Warmer Monitor.

Attributes of classes are simply the instruments belonging to the proto-class. The methods are drawn either from the component activities (as in the case of the Coffee Pourer class) or their constituent GDAs (as in the case of the Light Controller class).

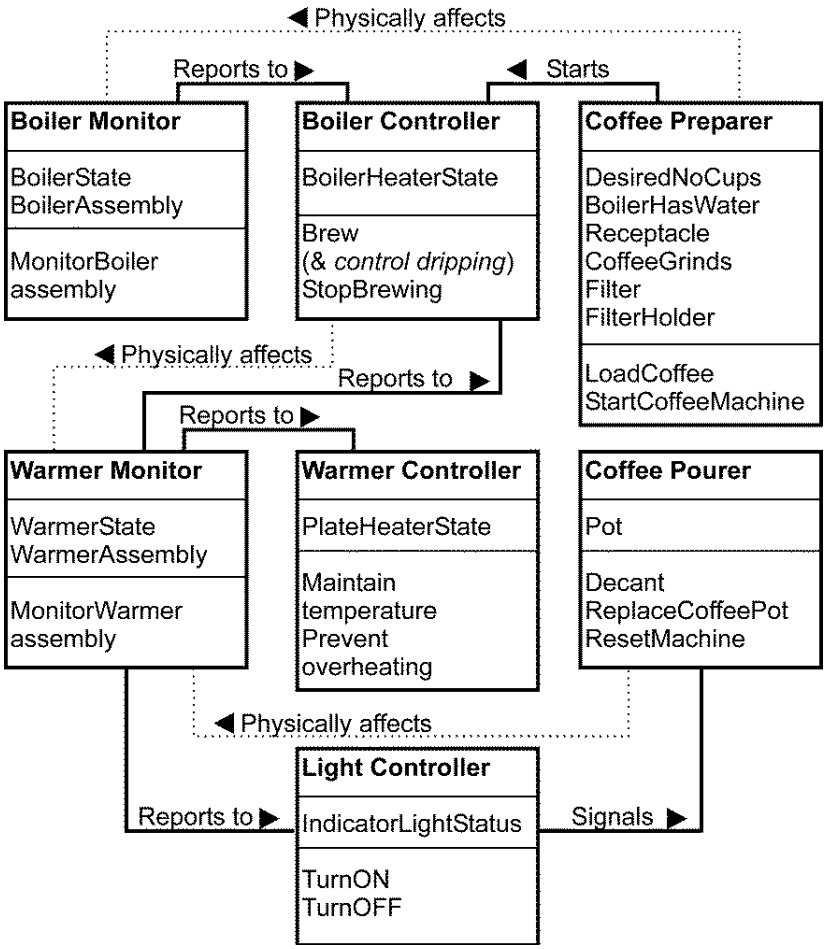

Fig. 2 Model-0 of the Coffee Maker offered by ATSA:OO.

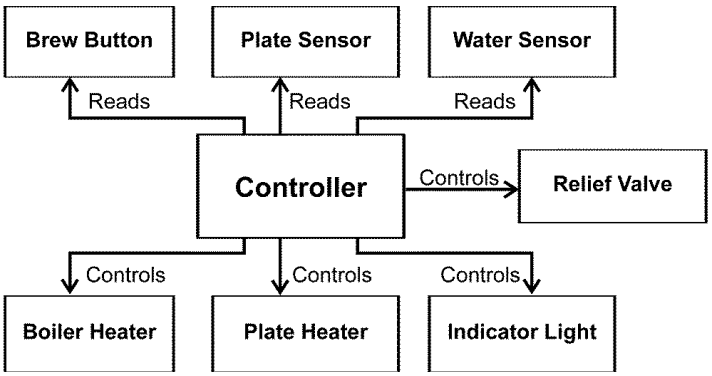

Fig 3. Model-0 of the Coffee Maker offered by traditional OOAD.(from Weirich 2004)

5.3 Model-0 Class Diagram

With the classes defined, as that remains is to determine the associations between the classes, and these can be directly inferred from the instrument transactions which cross the proto-class boundaries in the activity network. For example: the ButtonPress transacted between activities 2 and 6 indicates that the Coffee Preparer starts the Boiler Controller, while the WarmerState transacted between activities 11 and 6 suggests the Warmer Monitor reports to the Warmer Controller (assuming a subscription model). Our resultant model-0 is shown in Figure 2.

6 Comparison of ATSA:OO with OOA

A worked solution to the coffee maker problem was offered by Weirich (2004). Figure 3 shows his model-0, resulting from OOA Use Case techniques. It has a central ‘God’ class mediating between all the physical entities from the initial problem. Considerable effort would be required to reform it into workable classes.

The model-0 produced by ATSA:OO does not suffer from this problem. By fully decomposing to SINs then reclustering, ATSA separates the problem into distinct activities. Because the ATSA:OO heuristics identify the smallest feasible classes resulting from those activities, it is highly unlikely that a single ‘God’ class will emerge. This prevents a mistake common to neophyte practitioners.

Furthermore, because all activities are determined through clustering role and goal, the resulting suggested class methods are functionally cohesive. As the ATSA:OO heuristics cluster activities with their associated instruments each class is likely to have high informational cohesion (and therefore reduced coupling).

The OOA model-0 only has class names whereas the ATSA:OO model-0 has suggestions for possible attributes and methods.

Both the OOA and the ATSA:OO model-0 show class associations. However, neither depicts composition or inheritance, as these more abstract relationships are expected to be identified during the OOD phase.

7 Conclusion and Future Work

ATSA:OO is an adaption of the ATSA method to suit the widely adopted OO paradigm. It is intended to replace the OOA phase, with a method that is easier for neophyte practitioners to understand and use and has a strong theoretical basis.

If even an experienced industry practitioner, employing Use Case analytical techniques under OOA, can produce a model-0 with a controlling ‘God’ class, and

lacking both attributes and methods; then a neophyte practitioner can hardly be expected to use OOA with confidence. ATSA:OO's application of the precepts of AT yields classes with greater cohesion, ensuring that neophyte practitioners can have confidence in a stronger model-0, requiring less effort in the design phase.

There is substantially less reliance upon the intuition of the analyst, as ATSA:OO's model-0 arises from a straightforward application of simple steps. Each of these steps is sufficiently tightly prescribed to prevent see-sawing between potential models in the analysis phase.

ATSA allows the client to follow the design process all the way through to the activity network. ATSA;OO continues this philosophy by clustering the activity network directly, allowing the client to follow all the way to the class diagram.

This paper considered the feasibility of ATSA:OO using a very simple example. Further examination of this adapted method will consider more complex design problems. The clustering rules of both ATSA and ATSA:OO appear to result in highly cohesive classes; quantitative evaluation of the cohesion found in code built under ATSA:OO analysis is required to measure this. Similarly, quantitative and experiential tests with groups of neophyte practitioners will measure the ease with which the method can be learned and applied.

REFERENCES

Brown, RBK, Piper, IC (2011), What Users Do: SA&D with the ATSA Method, 20th Int'l Conf. on Information Systems Development (ISD2011) Edinburgh, Scotland 24 - 28 August 2011

Brown RBK, (2010), The ATSA Method for Specifying both System and user Interface Requirements, PhD Thesis, University of Wollongong, Australia.

Brown RBK, Hyland P and Piper IC, (2006), Specifying System Requirements Using the 5S Method ACIS 2006 Proceedings. Paper 100. http://aisel.aisnet.org/acis2006/100

Coad, P., & Yourdon, E. (1991). Object-oriented analysis (2nd ed.). Prentice-Hall.

Crear J (2009), CIO Standish Group. [online] http://www1.standishgroup.com/newsroom/chaos_2009.php [accessed 24th March 2010]

Engström Y (1987), Learning by Expanding: an activity-theoretical approach to developmental research, Orienta-Konsultit Oy, Helsinki, Finland.

Holmboe C. (2004) A Wittgensein Approach to the Learning of OO-modeling, Computer Science Education, 14:4, 277-296

Leont'ev AN (1978), Activity, Consciousness, and Personality, Prentice Hall.

Martin RC. (1995) Designing Object Oriented C++ Applications Using The Booch Method Prentice Hall; 1st edition

Stevens WP, Myers GJ and Constantine LL (1974) Structured Design, IBM Systems Journal 13:2 115-139.

Svetinovic D, Berry DM and Godfrey M (2005) Concept Identification in Object-Oriented Domain Analysis: Why Some Students Just Don't Get It, 13th IEEE International Conference on Requirements Engineering (RE'05)

Vygotsky LS (1978), Mind in Society Harvard University, Press, Cambridge, MA.

Weirich J. (2004) OOAD Design Problem: The Coffee Maker [online] www.csun.edu/~renzo/cs680/coffeebrewerdesign.pdf [accessed 20th February 2012]