(GROUP LEADER) – Rui Kuang (rkuang@cs.columbia.edu)
Andrey Butov (ab2301@columbia.edu, andreybutov@yahoo.com)
Arvid Bessen (ajb2103@columbia.edu, bessen@cs.columbia.edu)
Svetlana Starshinina (svs29@columbia.edu)

Wizard of ***HO***use ***M***anagement (WHOM)

The WHOM language is designed to allow developers to simulate the creation and management of a household. Using WHOM, it is possible to define a household as well as a set of rules to govern the automatic operations of that household.
WHOM is designed to be a simple but powerful, object-oriented, event-driven, flexible, and robust language.

Overview

Many sci-fi writers envision a future world in which most of the household tasks would be performed by machines. Even today many objects exist that have the capacity of being programmed. Examples include VCRs, alarm clocks, etc. It is not so hard to imagine that in the future many household objects, from windows to full-featured security systems will come with built-in computers capable of controlling various built-in functionalities. A simple window, equipped with the right mechanism, can be easily opened or closed, and locked or unlocked automatically without requiring any action by a human. The possibilities are virtually endless, and could involve not only windows that open and close, but doors that lock and unlock, and refrigerators that automatically restock themselves by contacting the nearest supermarket and ordering the necessary foods to be delivered at a certain time. With a slight stretch of imagination, it is also possible to imagine that in the future objects would be able to perform more complicated tasks, such as windows washing themselves, beds making themselves, and entire households being able to implement internal security and ambiance features. We have assumed that our objects are “futuristic” with complex mechanisms that enable them to perform a variety of function. All objects in the house are connected to a single Household Management System (HMS), which allows them to communicate.

The WHOM language allows to define the behavior of these objects. Instead of issuing commands to each of these objects, the WHOM language is intended to provide a uniform and easy-to-use programming interface to the objects in the house. This way the objects will be able to act even when the owner of the house is not present at all. Thus, rather than going into the trouble of issuing direct commands, the user would be able to write a program in the WHOM language to specify what the objects should do, and in response to what events.

Considering the fact that a majority of the population has no programming experience, we had to make the language very simple to use. In order to use the language, first the house itself needs to be defined and then all the objects inside the house need to be “plugged in”. This may be done by a WHOM system programmer, similar to how some things get installed by the electrician currently. However, the language is simple enough for the user, to be able to define behavior by first issuing simple commands (such as “close window when temperature < 70”), which can be combined to compose a more intelligent household management system.

The key characteristics of the language are outlined in greater detail below.

Simple:

WHOM is very easy to use for users, even those that have minimal programming experience. The commands are very close to the English language and the syntax is very intuitive. The language provides several built-in classes, which can be very easily manipulated by the user by issuing commands such as “close”, “lock”, etc. The object status is periodically examined and appropriate functions can be performed to change the status of the object. Behavior intrinsic to the various objects allows the user to be able to perform simple household tasks with great ease. Additionally the language supports the measurement of several environmental variables, such as temperature, light intensity, etc, and allows the user to specify the appropriate preset or customized response to changes in the environment.

```
// define house
Window northWindow;
Window southWindow;
Window eastWindow;
Window westWindow;

all(Window).close;

Group somewindows = { northWindow, eastWindow, westWindow };
somewindows.open;
```

Powerful:

WHOM is very powerful even for beginners. For more advanced users WHOM presents virtually no limits. In addition to the built-in classes, the users can specify their own classes, with their own set of parameters and commands. Adding a new object type to the household management system will be equivalent to including a library class in a Java program, and then all the commands to manipulate the object will automatically become available to the user. The users can also create new functions in order to simplify more frequent tasks. It is possible to program the HMS to respond to such complicated events like when the kids come home from school,, or to program the system so that the house cares for itself while the family leaves for vacation.

```
// define house
Window northWindow;
Window southWindow;
Security alarm
```

```
if ( any(Window).broken )
{
     alarm.activate;
     phone.callPolice;
}
```

Object-oriented:

The language is heavily object-oriented, since most of the commands in WHOM are commands to objects to perform certain actions. In order for an object to execute a command, it has to be defined for that object. If a certain command does not exist, an error message will be issued. The various objects in a system will be "plugged-in" into the house management system. The objects will be either standard and predefined or customized and defined by the user, similar to existing libraries and self-defined classes in Java. The standard objects, such as windows, doors, TVs will have a predefined set of actions that could be performed on them. The customized objects may come with a more complex set of actions that need to be specified by the user.

Extendible:

The user can specify any kind of house, with any number of rooms, windows, doors and other objects. Any object can be added or removed from the home management system at any time. The actions performed by objects may be scheduled or event driven as desired by the user. It is possible to expand the functionality greatly by introducing third-party libraries such as a home-bot for performing more complex actions with other objects. This home-bot would only need to be specified as an object that is able to perform a great variety of actions using various objects, with a specified movement algorithm. It is also possible to create various preset modes (to help the user using the new household management system) for buildings such as prisons, family homes or kindergartens.

```
class FancyDoor extends Door
{
Variables:
     Boolean catDoorOpen;

Constructor:
     FancyDoor;

Methods:
     playMusic;
     openCatDoor;
}

class Cat extends Pet
```

```
{
     // …
}

if( any(Cat).scratch( FancyDoor d ) )
{
     d.openCatDoor;
}
```

Event-driven:

The entire household management system will be event-driven, with a capability of responding to complex events such as a person entering the room, a robber trying to break into the house, or kids coming home from school. We assumed that some of the objects would have complex movement-detecting, face scanning mechanisms to be able to respond to these kinds of events. For example, the home management system will never limit the actions of the adults in the house, but it may limit the actions of the children (not let them watch TV if they did not do their homework). The events will serve as triggers for the objects to perform a specific kind of action, which could involve performing a direct action, or sending some specific information both to the user or to an emergency telephone number (such as 911)

Concurrency:

WHOM will allow concurrent events to take place. For example, if all windows have to be closed in response to a sudden thunderstorm, it would be inefficient, if the language first sent the signal to one window, then the other and so on. Instead the signal would be sent simultaneously to all windows and they will respond with the appropriate action (closing).

```
Thermometer t;
if( t.temperature < 80 )
{
     // Close all windows simultaneously
     all(Window).close;
}
```

Robust:

WHOM allows for writing programs that are simple, with extensive error checking being done both at compile-time (by our WHOM compiler and javac) and at run-time. If the user attempts to issue a command to an object that does not exist, he/she will be informed at compile time. If the user attempts to issue an invalid command to an object he/she will be informed at runtime by a message from our simulation system.

List of sample built-in objects and actions

Objects	**Action**
Window	Open/close, lock/unlock, clean
Door	Open/close, complex action which responds to a person trying to enter, depending on who the person is
Fridge	Dispose of an item, stock the fridge (by ordering)
Lamp	Turn on/off
Floor	Wash, sweep

Implementation Plan:

The front-end of WHOM, which produces intermediate code for the back-end, will be implemented using ANTLR and Java. Java will also be used for the back-end, which is responsible for translating the intermediate code into an executable. The simulation part will either be integrated in the back-end part and put along with the translated Java code or be a separate program, which communicates with the WHOM program for simulation and testing. If it is in a separate program, Java (possibly some other languages convenient for graphics) will be used.

Conclusion:

We outlined WHOM, a new object-oriented, event-driven, concurrent, and robust house behavior programming language. As shown by our examples, WHOM allows to write readable and powerful code, even for novices, while at the same time being the ideal tool for the power user with advanced needs, because of its easy extendibility.
In the near future WHOM, with these features, will probably become the most popular and widely used house behavior programming language.