End User Requirement Specification for Spreadsheets

Laura Beckwith
Oregon State University
beckwith@cs.orst.edu

End user programming has recently received a lot of attention, but there has been little research into aspects of end user programming beyond coding. Coding is only one part of the development process; focusing on other aspects could possibly make more reliable programs. Reliability is an issue in end user programming, as shown by statistics about spreadsheets, a widely used type of end user programming languages with years of history. Studies done on spreadsheets have found a disturbing number of spreadsheets that have errors: a conservative range is that 20%-40% of spreadsheets have errors.

Perhaps spreadsheet reliability can be improved if the spreadsheet users work collaboratively with the computer to communicate the relationships and possible values to the system. We suspect spreadsheet users know more about the purpose and underlying information of the uses of the spreadsheet than they are able to communicate to the system. Our goal is to allow end users to communicate with the system about their requirements. This allows checks and balances, so that if the spreadsheet does not conform to the requirements, the system can call this to the users attention. "Requirements specification" (RS) is the term traditionally associated with this concept by the software engineering community. But, can end users deal with requirements specification?

We are pursuing this question using the research spreadsheet language Forms/3 as a vehicle. Other than the work done in Forms/3, we have been unable to locate any published work on making RSs available to end users. To try to address this lack, other members of our Forms/3 research group have implemented preliminary work on the user interface and propagation of RSs on simple cells, but the types of spreadsheets investigated so far are tiny with only a few simple cells. Within this small prototype we have been able to deal with ranges and Booleans.

Whenever a user puts an RS on a cell, whether through taking the initiative to do so or as a by-product of a conversation with the system about right and wrong values noticed, that RS is propagated through formulas downstream generating system-generated RSs on each downstream cell. A cell that has both a system-generated and user-entered RS is in a conflict state if the two RSs do not match exactly. As Figure 1 shows, to communicate conflicting multiple RSs on one cell, the system circles the icons of the conflicting RSs. The stick figure icons shown in Figure 1 represents each cell's user-entered RSs. The computer icon represents cell Celsius's system-generated RS. In Figure 1, the simple cell Celsius has an RS conflict. The scenario leading to this conflict is that the user first entered formulas then entered RSs on both cells, the RSs were not consistent with the propagated RSs. If the value produced by the formulas had been inconsistent with the RSs, it too would be circled.

The main goal in this research is to establish scalable RS mechanisms that are viable for end user spreadsheet programming involving large spreadsheets. In order to be viable, the mechanism must be able to seamlessly integrate RSs within spreadsheets, reason about propagating RSs, and in doing the propagation must be efficient enough for the immediate visual feedback demanded by spreadsheets. Most important is our focus on how end users will work with the RSs in spreadsheets. Thus, a significant portion of this research will involve empirical studies conducted with end users. Preliminary empirical work in this direction has already been started in our group.

Most large spreadsheets consist of grids with repeated formulas. In Forms/3 grids can be sized to any number of rows and columns of cells. The rows and columns can then be broken into subsections, referred to as "regions". A region is a group of cells that share one formula; in Figure 2 the three cells in the column labeled "Quiz Avg" are a region. Other spreadsheet languages such as Formulate and Lotus also use the region concept, and the same functionality can be found in Excel and similar languages through formula replicate. In Figure 2, the first two columns are the values the students received on the quizzes, and the third column computes the average of the quiz scores in the first columns. Notice that there is just one formula for all three cells; this is an example of a region formula. Grids can be large, and generally are what make spreadsheets large.

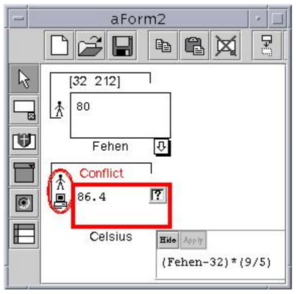
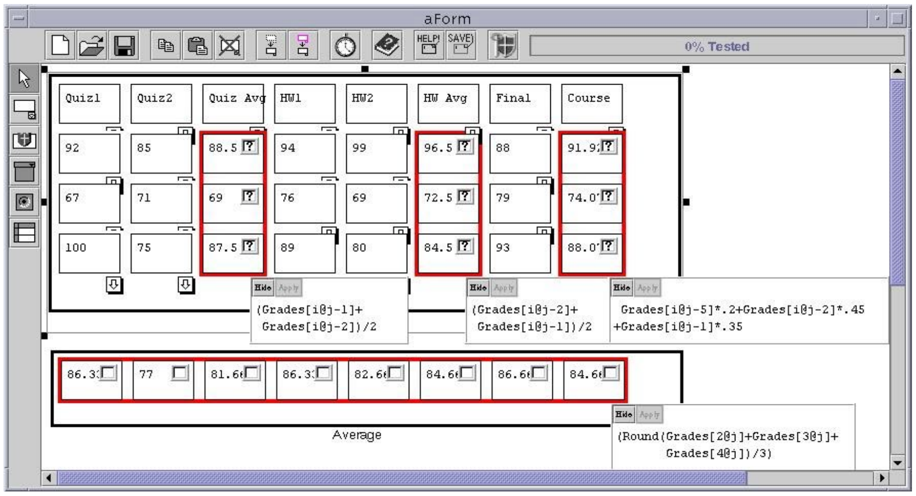


Figure 1: A Fahrenheit to Celsius conversion.

The table lists some of the problems we need to consider in this research. Due to a lack of space only one of these will be discussed in detail.



Quiz1	Quiz2	Quiz Avg	HW1	HW2	HW Avg	Final	Course
92	85	88.5	94	99	96.5	88	91.9
67	71	69	76	69	72.5	79	74.0
100	75	87.5	89	80	84.5	93	88.0

```
(Grades[i@j-1]+
 Grades[i@j-2])/2
```

```
(Grades[i@j-2]+
 Grades[i@j-1])/2
```

```
 Grades[i@j-5]*.2+Grades[i@j-2]*.45
+Grades[i@j-1]*.35
```

86.3	77	81.6	86.3	82.6	84.6	86.6	84.6

```
(Round(Grades[2@j]+Grades[3@j]+
       Grades[4@j])/3)
```

Figure 2: Two grids: Grades (top) computes grades for each student, and Average (bottom) averages each column in Grades.

The table's first entry describes the problem of how to handle multiple user-entered RSs on one cell. Until now we allow only one user-entered RS per simple cell. However, multiple user-entered RSs seem necessary in grids. For example, a user may need to specify a range RS on a row and another on a column. This RS would overlap on at least one cell. A second possibility could arise in entering a range RS and a "relationship RS". A relationship RS would allow the user to specify how two cells (or groups of cells) should be related, as a cross check to the formula. For example, in Figure 2 the user could put a relationship RS on the rightmost cell of the Averages grid, specifying that the Course column's formula, which computes each student's course grade, could also compute the overall average course grade when applied to the Average grid. This specification would crosscheck Average's formula, which instead computes the last column as the average of the Course column. Both ways should obtain 84.66 in the last cell. Such multiple RSs give users more ways to enter checks and balances.

The problem is, how to define the notion of these multiple RSs being in conflict. The idea of "must exactly match" used in our preliminary work will no longer suffice. For example in Figure 2, we can imagine the user of the spreadsheet wanting to know when a particular student's grades fall below a 70. To do this the user puts as an RS on row of that student, a range RS of 70-100. If there is also an RS on the columns that all grades are between 0-100, this student has multiple user-entered RSs on all cells in this student's row.

One possibility is to allow one RS to be a subset of another without conflict. However, this would prevent the system notifying the user of inconsistencies among their RSs. The above student example was an inconsistency the user wanted; on the other hand, it could instead have been a mistake, and for the system to simply ignore that inconsistency would diminish the power of the system to provide feedback about errors.

So, either decision made by the system is incorrect for some cases. Another possible approach is to have the user make the decision. If the user makes all the decisions, such as if two RSs should match exactly or if subsets are acceptable, the benefit is the decisions are the ones the user wants, but the cost is the time the user must spend doing the deciding. Having the system make the decision to cut this cost runs the risk of later cost due to potentially bad decisions (c.f. Blackwell and others in "Attention Economics"). Even more important, the user needs to believe they understand how conflicts arise, the importance of these conflicts, and what can be done to resolve them, because research shows that for users to trust a system they have to believe they understand how it works. One way of encouraging user understanding is by requiring them to make the decisions. If we proceed in this direction, we must do so in a way that does not demand the immediate attention of the user. We are able to avoid demanding their immediate attention by using non-intrusive devices, such as changes in markings that can be attended to as the user desires. Thus finding the balance between allowing the users to make the decisions when they want, and making decisions for them is a critical part of this research.

Problem	Discussion
Multiple Relevant RSs	How do we define conflicting RSs? What does it mean to have RSs that supplement each other?
Seamlessness	Must be consistent with spreadsheets, end-user programming and the WYSIWYT methodology.
Laziness	Does RS propagation need to be eager? If so, does that mean that the language cannot use lazy evaluation?
Efficiency	Choices of how to solve problems must be efficient enough to maintain immediate visual feedback.
Reasoning about input areas	Option 1: Generalize RSs on all input cells to one RS via inference and/or Option 2: Allow user to select an input area to contain the same RS across selected area.