WD-PIC Experiences

Let me describe 1 other experience of using the UM for a program as its RS.

In this case, the program has a graphical user interface (GUI). Thus, the manual cannot be input directly to the program, and thus, the manual itself cannot be used as a test case for the program.

WD-PIC, a WYSIWYG Direct-Manipulation PIC

Faina Shpilberg
Daniel M. Berry, 1998

WD

“WD” stands for “WYSIWYG, Direct manipulation”

“WYSIWYG” stands for “What You See Is What You Get”

They are independent in principle, but usually come together in WIMP (Windows, Icons, Menus, Pointers) interfaces.

Batch vs WD Drawing Programs

Batch, e.g., PIC:

A can edit PIC specification with batch editor; insertion, global changes, working with groups, more convenient than with most WD drawing programs

D cannot see what you are doing

WD, e.g., xfig:

A can see what you are doing

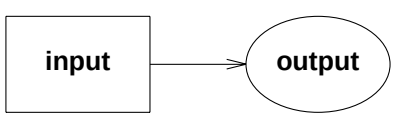
D painful to make insertions and global changes and work with groups

PIC

The input:

```
box "input"
arrow
ellipse "output"
```

yields

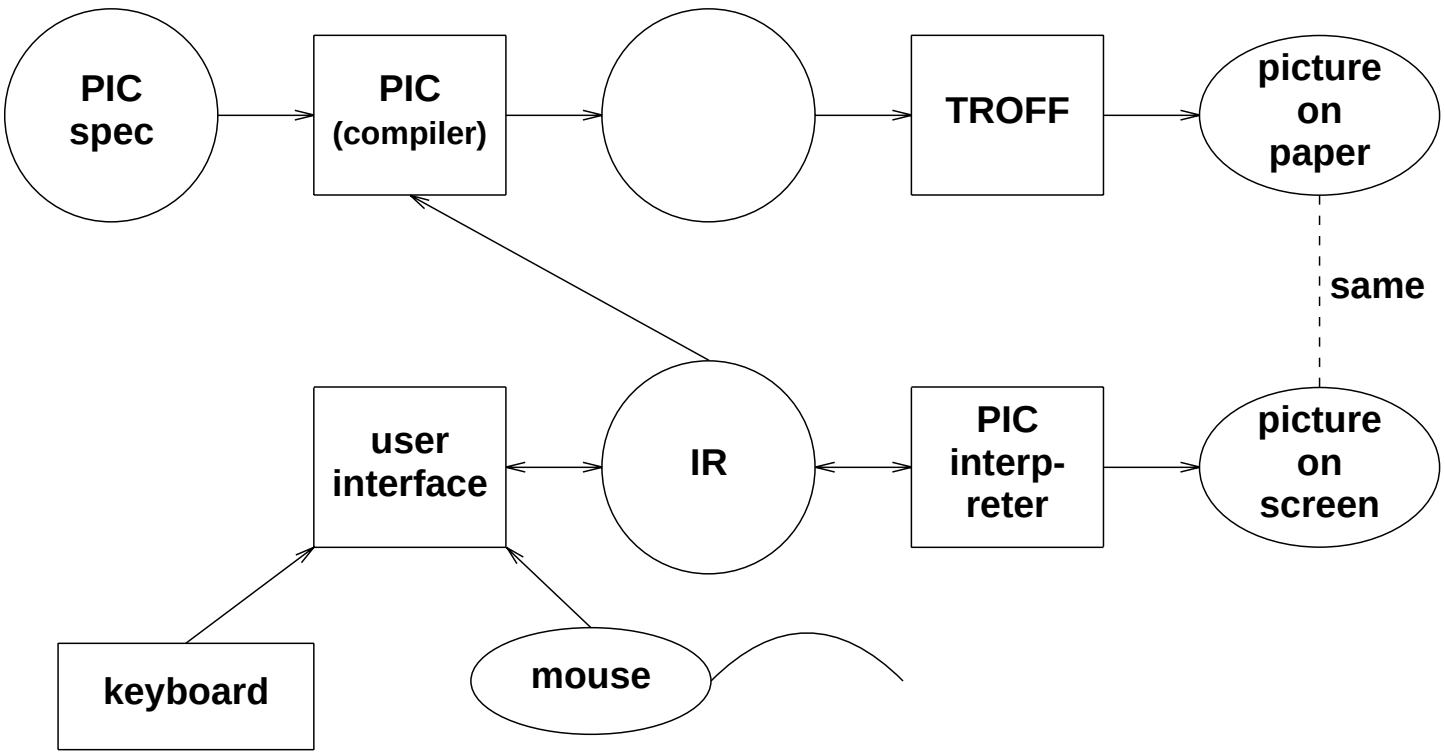


Goals of WD-PIC

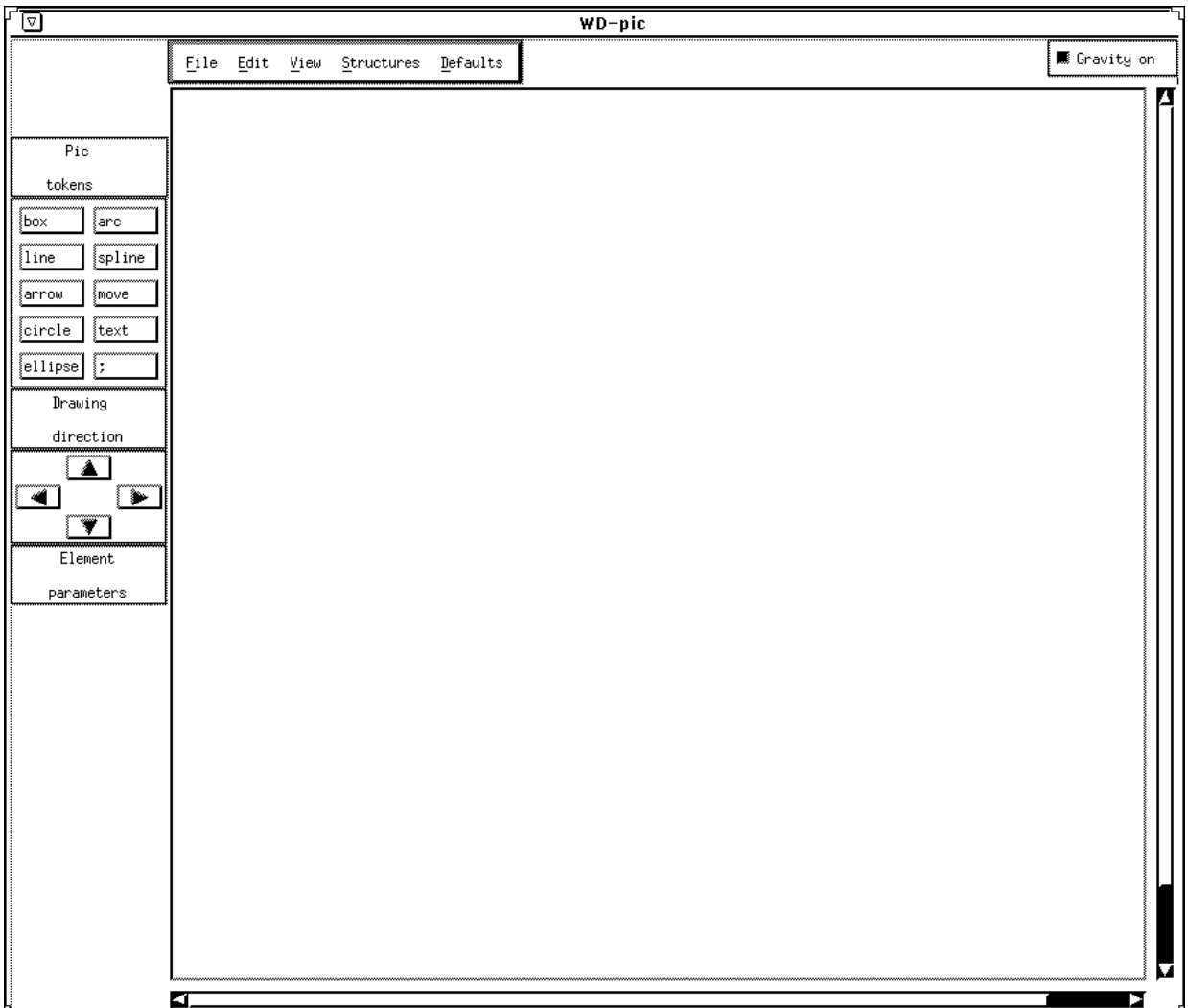
BOBW (Best of Both Worlds):

- **editable internal representation (IR), in the PIC language,**
- **can see the picture being drawn on the canvas,**
- **at any time the IR is a PIC specification of what is on the canvas,**
- **palette of PIC elements and menues for their attributes, and**
- **pull down menues for files, editing, views, and others.**

Batch PIC



WD-PIC Flow


WD-pic
File Edit View Structures Defaults
Gravity on
Pic
tokens
box
arc
line
spline
arrow
move
circle
text
ellipse
;
Drawing
direction
Element
parameters

WD-PIC vs. other WD Drawers

Because of equivalence of batch and WD picture for any IR,

in WD-PIC, rarely have to move mouse to canvas;

can stay in palette, clicking away.

In other WD drawers, must move mouse to canvas to position a clicked palette item.

Requirements -1

At any time, picture on canvas is same as printed on paper when the accumulated IR is submitted to PIC|TROFF.

At any time, clicking on *X* box is the same as entering "*X*" on the keyboard, for any PIC element *X*.

Requirements -2

E.g. all of the following are equivalent:

(stand-alone character is typed, labeled box is clicked)

b o x " i n p u t " ↵ a r r o w ↵

box " i n p u t " ↵ arrow

b o x " i n p u t " ↵ arrow

Requirements -3

Avoid dialogue boxes and confirmation buttons.

Cut, copy, paste, and other direct manipulation at the graphic level.

Requirements -4

IR of picture should be what human being would type, making use of defaults, and not showing full parameters with 8 digit floating point numbers as parameter values, e.g.:

box "input"

rather than

box wid .75237589 ht .58639282
at 3.8203785, 2.9851863 "input"

Requirements -5

(Note how these last two can contradict each other; moving a box to an arbitrary point requires pairs of 8 digit floating point numbers as coordinates.)

Grid of symbolic distances with origins in symbolically identifiable places, e.g., movewid $\times$ moveht grid centered at Box.ne.

Requirements -6

Can point to a grid point in order to, e.g., put things there by DM.

Can, at any time, edit the IR with the text editor associated with $EDITOR.

When save and exit from text editor, the picture is regenerated in the canvas.

These are the basic goals and requirements.

Prototypes

Several prototypes over several years for
- a MS thesis
- capstone projects
- class projects

First Production Version

Lihua Ou took the assignment to produce a first production-quality version of WD-PIC as her master's thesis project.

Ou's Professional Background

Prior to coming to graduate school, Ou had built other systems in industrial jobs, mainly in commerce.

She had followed the traditional waterfall model, with its traditional heavy weight SRS.

She had made effective use of libraries to simplify development of applications.

Ou's Input

Ou was to look at all previous prototypes and UMs as specifications.

She was to filter these and scope them to first release of a production quality first version of WD-PIC running on Sun UNIX systems.

Ou's Assignment -1

Ou was to write a specification of WD-PIC in the form of a UM.

This UM was

1. **to describe all features as desired by the customer, and**
2. **to be accepted as complete by the customer,**

before she began any design or implementation.

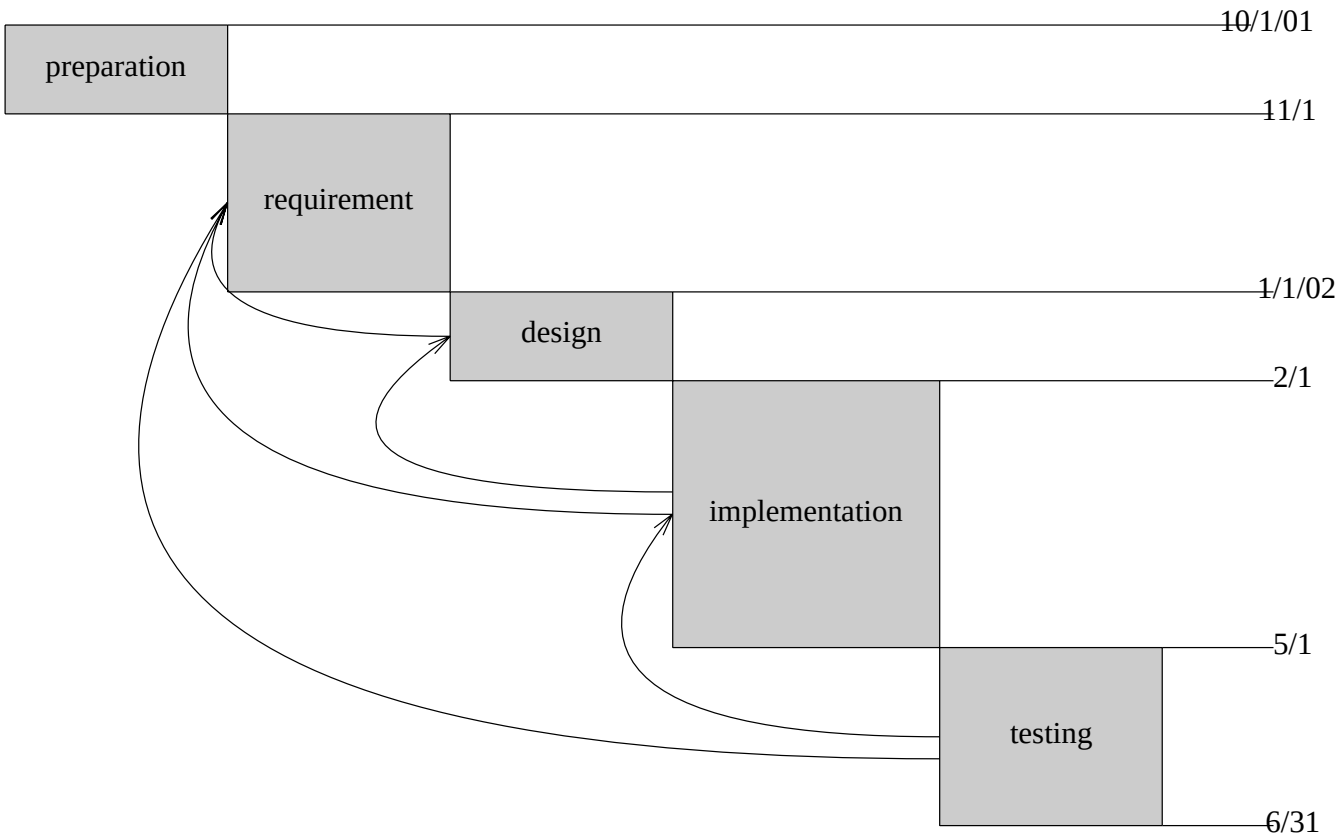
Ou's Assignment -2

Once implementation started, when new requirements are discovered, the manual should be modified to capture new requirements.

In the end, the manual describes the program as delivered.

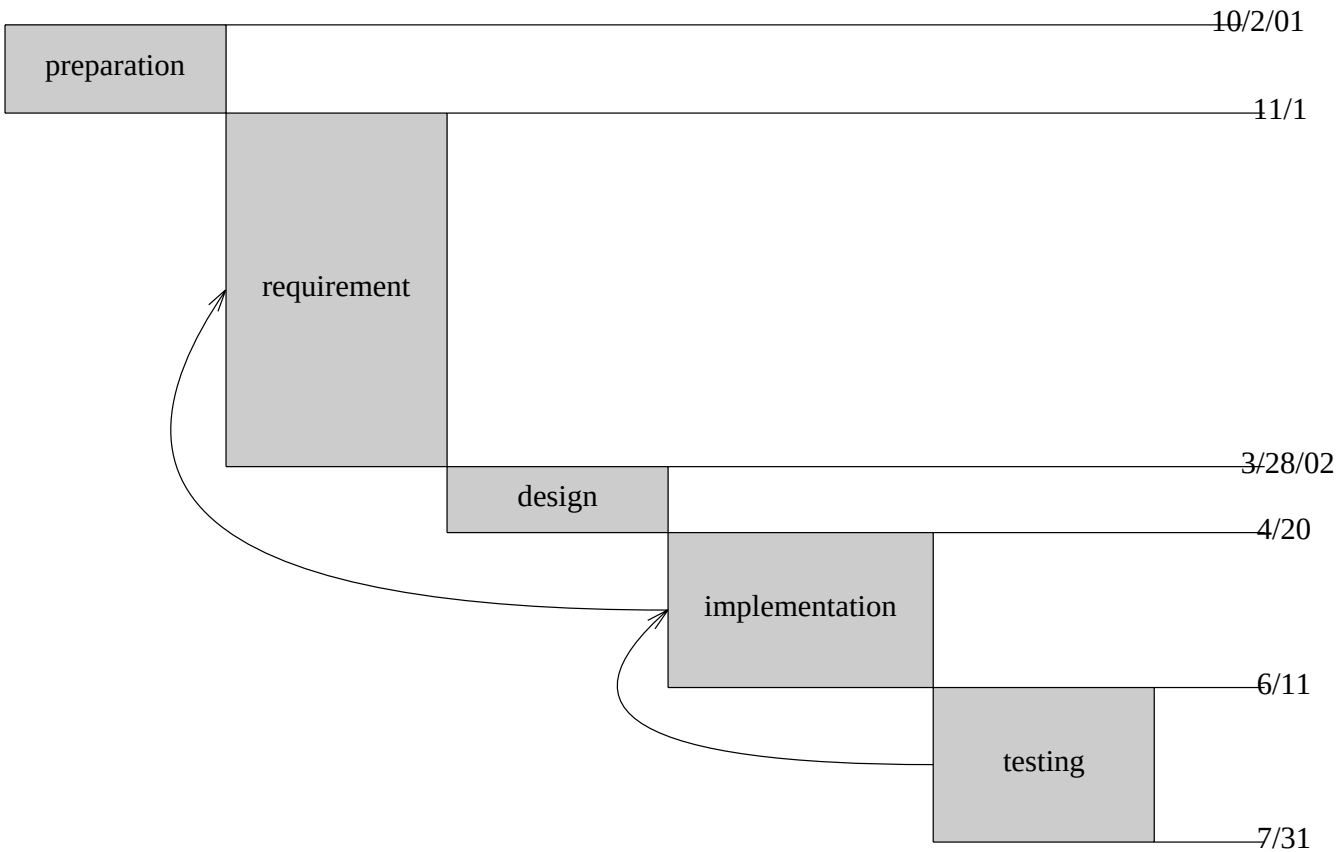
Project Plan

Duration in months	Step
1	Preparation
2	Requirements specification
4	Implementation
2	Testing
1	Buffer (probably more implementation and testing)
10	Total planned


preparation
requirement
design
implementation
testing
10/1/01
11/1
1/1/02
2/1
5/1
6/31

Actual Schedule

Duration in months	Step
1	Preparation
4.9	Writing of user's manual = reqs spec, 11 versions
.7	Design including planning for maximum reuse of PIC code and JAVA library
1.7	Implementation including module testing and 3 manual revisions
1.7	Integration testing including 1 manual revision and implementation changes
10	Total actual


preparation
10/2/01
11/1
requirement
3/28/02
design
4/20
implementation
6/11
testing
7/31

What Happened?

While detailed plan was not followed, total project time was as planned.

Also, Ou produced two implementations for the price of one, for:

- **(planned) Sun with UNIX and**
- **(unplanned) PC with Windows 2000**

Surprise

Ou was more surprised than Berry that she finished on time.

Berry had a lot of faith in the power of good RE to reduce implementation effort.

Adding to Ou's surprise was that the requirements phase took nearly 5 months instead of 2 months; the schedule had slipped 3 months out of 10, way beyond recovery.

Then and ...

Ou's long projected implementation and testing times and the 1 month buffer indicate that she expected implementation to be slowed by discovery of new requirements that necessitate major rewriting and restructuring.

Then and Now

This time, only minor rewriting and no restructuring.

Thus instead of 2 months specifying and 7 months implementing and testing,

she spent 5 months specifying and only 4 months implementing and testing.

Why?

By spending 3 additional months writing a specification that satisfied a particularly hard-nosed customer who insisted that the manual convince him that the product already existed,

Ou produced a specification that

- **had very few errors and**
- **that was very straightforwardly implemented.**

The Errors

Almost all errors found by testing were relatively minor, easy-to-fix implementation errors.

The two requirement errors were relatively low level and detailed.

They involved subfeatures in a way that required only very local changes to both the manual and the code.

What Helped?

All exceptional and variant cases had been worked out and described in the manual.

Thus, very little of the traditional

- **implementation-time fleshing out of exceptional and variant cases and**
- **implementation-time subconscious RE.**

Test Cases

The manual's scenarios, including exceptions and variants turned out to be a complete set of black box test cases.

Tests were so effective that, to our surprise, ...

scenarios not described in the manual, but which were logical extensions and combinations of those of the manual worked the first time!

The features composed orthogonally without a hitch!

Satisfied Customer

Berry found Ou's implementation to be production quality and is happily using it in his own work.