

Prototyping

Outline

Prototyping in the Life Cycle

Varieties of Prototypes

Practical Experience

Implications for InfoSys

Walkthroughs

see also Lichter, Schneider-Hufschmidt, and Zuellighoven,
Prototyping in Industrial Software Projects, *IEEE Trans. Software
Engineering*, Nov. 1994

Terminology

According to Webster:

pro-to-type $\acute{p}ro\text{-}t\text{-}e\text{-},t[0xF5]\text{-}p\ n$
[F, fr. Gk $pro\text{-}totypon$, fr. neut. of $pro\text{-}totypos$
archetypal, fr. $pro\text{-}t\text{-}$ + $typos$ type]
1: an original model on which something is
patterned: ARCHETYPE
2: an individual that exhibits the essential
features of a later type
3: a standard or typical example
4: a first full-scale and usu. functional form of
a new type or design of a construction
(as an airplane)

In software engineering:

- different specifics
- same goals

The lighter side:

“prototype” is just another name for a
project which failed

“Buying Information”

Projects have

- ★ uncertainty
specification & design present many difficult decisions
- ★ risk
 - † wrong functionality
 - † unusable interface
 - + many, many others

Need information to avoid risks and facilitate decisions

∴ invest in information gathering thru experimentation

Q: Would it be wrong for a prototype to use most of the project budget?

A:

Relation to Life-Cycle Models

Prototyping

Goal: answer specification & design questions

- ◇ fits in any life-cycle model

Phased delivery

Goal: varies, usually early delivery of partial product

- ◇ requires special adaptation of life-cycle model

Research

Goal: technology or approach embodied in prototype

- ◇ out of typical lifecycle
- ◇ most university projects

Prototypes & Projects

A project may have many prototypes

Different kinds of projects call for different kinds of prototypes

IS projects have UI prototypes

Target may be

- product
- process

Classification

details discussed on following slides

Presentation

Experimentation

- breadboard
- mockup
- production prototype
- pilot

Phased delivery

- pilot
- evolutionary development
a.k.a. “rapid prototyping”
- iterative enhancement
- ✗ contrasts with “throw-away prototyping”

Presentation Prototype

- marketing or marketing-related
- goal is not to discover information but to convey it
- does not fit with “information buying”

Experimentation

Breadboard

Q: will it work?

Q: does it have the desired characteristics?

e.g. automobile handling

- functionality

Mockup

Q: will it be usable?

e.g. airplane cockpit

- appearance
Human-Computer Interaction (HCI)

Production prototype

Q: can it be mass-produced?

not relevant for software

Pilot

Q: can it be operated?

Exploratory Prototyping

Horizontal

- only specific layer(s)

Vertical

- complete implementation of subset of functionality

Building blocks

- may be used in prototype development
- may turn prototype into component

Phased Delivery

Pilot

- delivery to a few sites

Evolutionary development

- deliver wide but partial functionality
- “layer at a time”

Iterative enhancement

- deliver few components with complete functionality
- “slice at a time”

Voices of Experience

“Plan to throw one away; you will, anyhow.”

– Brooks, *The Mythical Man-Month*

Study of prototype use in industrial software projects:

– Lichter *et al.*, *IEEE TSE* 20, 11

- five projects, multiple prototypes in each
- 2 to 240 work-years
- following slides report results

Experience - Negatives & Positives

- traditional project control techniques hinder prototyping
- contractual obligations and prototyping rarely fit together well
- effort for end-user involvement in prototyping usually underestimated
- prototyping cannot mitigate (and sometimes may amplify) problems of poor project management
hidden agenda
- + breadboarding facilitates more creative approaches

Experience - Suggestions

- ★ use prototyping as part of development strategy
planning must budget “buying information” through prototypes

validate functional requirements early
- ★ choose the right kind of prototype
turning a presentation prototype into a building block is a sure path to disaster
- ★ focus prototyping effort
ask one question at a time
- ★ involve end user
in spite of the “That’s great, let’s put it online tomorrow” phenomenon

Additional Objective – P465/P565

- design crucially dependent on medium
 - lack of knowledge about tool(s)
creates substantial schedule **risk**
- + many team members not familiar with the tool



- ☆ Prototype must include learning the tool !

Required Prototyping

	Look & Feel	Function-ality	Tool Exploration
Form			
components	✓	✓	✓
layout	✓		✓
navigation		✓	✓
Report			
layout	✓		
computation		✓	✓
Events			
explicit		✓	✓
implicit		✓	✓
form control	✓	✓	✓
data manip.		✓	✓
programmed		✓	✓
Constraints			
data validation		✓	✓
Database			
referential		✓	✓
integrity			
triggers			✓

Prototype Deliverables

on-line input:

- all core screens with click navigation but without functionality
- example screens with functionality, down to the bottom level for one major branch of the menu selection hierarchy (i.e. a depth-first exploration of a menu path)
- examples of data-entry screens
- events that are:
 - ◇ explicit (e.g. button-click)
 - ◇ implicit (e.g. entering a field)
 - ◇ form related (e.g. switching subforms, drop-down lists)
 - ◇ data manipulation related
 - ◇ implemented by program code, including “behind the scene” communication between code modules

data:

- all tables, essential attributes, not necessarily all constraints

on-line output:

- sample screens generated by major categories of queries

off-line input:

- copies of all forms

off-line output:

- draft layouts of major reports

validation of:

- at least one input value by format or domain (e.g. “integer from 1 to 100”)
- at least one input value as foreign key constraint

mandated interactions with other systems (where possible):

- demonstrate ability to connect and interact

error handling (beyond validation failures):

- trapping at least one database interface error
- trapping at least one user interface interface error

Walkthroughs – Formal Reviews

What:

- structured presentation of technical issues

Why:

- understand
- proactive QA – catch errors
- improve team performance – reduce rework
- project planning

Technical presentation ⇒

- not sales
- not management

Requirements Walkthroughs

- Goals – briefly
- Context & scope – briefly
 - ◇ interaction with other systems
- Workflow to be automated
- Information model
- Other functionality
- Qualitative requirements
 - ◇ only when some requirement stands out

Walkthrough Reviewers

Reviewers:

another team specifically responsible for critiquing

Deficiencies: identify aspects that are

- ★ unclear
- ★ ambiguous
- ★ lacking essential details

Particulars: where does team need to

- ★ enhance scope limit statements
- ★ clarify information model
- ★ generalize information model
- ★ facilitate workflow