Electronic Forms and XML

Shanti Rao

Raosoft, Inc.

www.raosoft.com

Electronic Form System

www.electronicform.org

Stay tuned for

eForm practices

Social engineering

eForm XML schema

eGov & complexity

The 21st century

Flying cars

Videophones

Robot housekeepers

Instant meals

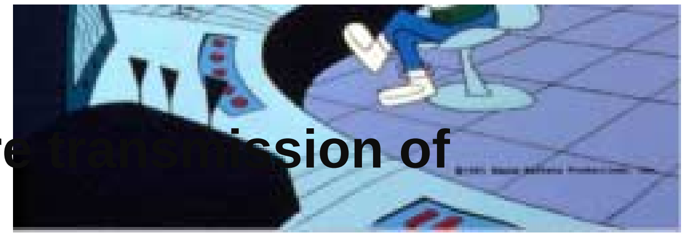
Paperless office

Objectives of the ~~paperless~~ office

Reliability

Information

History



Forms structure transmission of information

- **Strong structure** *checklists*
- **Weak structure** *Al Gore’s medical claim form*

Structure aids submittal and processing

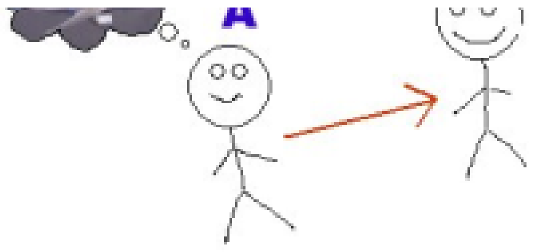
Purpose of forms

Provide information *Who owns jumbo jets?*

- **Regulation (reliability)**
- **Decision support (information)**
- **Statistics (history)**

Request for action *Alice wants to buy a jumbo jet*

- **Instructions for others (Bob)**
- **Assign responsibility**



Properties of forms

Encapsulation

- Instructions for Alice
- Organize related information for Bob

Archival value and record keeping

- Searchable fields
- Once it’s written down, you can forget it!
- Statistics

Consistency

- Forms become a business process
- Make copies & re-use

Commitment

- Digital signatures?

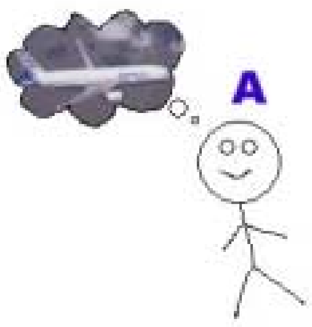
Example

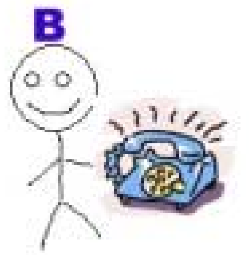
Alice *requisition form (request for action)*

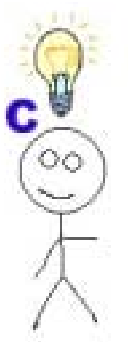
Bob *contact vendor (action)*

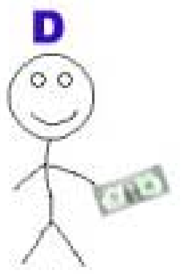
Cynthia *gives approval (decision)*

Dan *make payment (record keeping)*





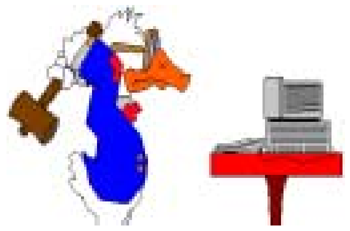




Key steps

Collaboration

Forms may be completed over time
Peer review and correction
Preliminary submission ("Is this right?")
Cooperation ("Just leave that line blank.")



Validation

- Avoid complex logic tests
- Calculations on server, not client

Gather information

I. Many people send information to one person

- Tax returns

II. Many people collect the same information

- W-2 form

Record a decision

III. Forms identify a series of decisions

- Purchasing

What you need to know about a decision

How

Who

What

When

Sometimes why

What you need to get it

Who has the answers?

Design recapitulates bureaucracy

Information owned by many people

Who *Alice, Bob, Cynthia, Dan*

What *Alice, Cynthia*

When *Cynthia, Dan*

How *Bob, Dan*

Dangerous ’workflow’ techniques

Make a form for each stage in the business process

Effectiveness depends on understanding interactions

Capital cost stifles innovation

George Jetson

"We do not anticipate any unforeseen obstacles."

Best practices

Design

KISS

- Expect the unexpected
- Minimize reliance on client software

Social engineering

- Provide support, not restraint
- Encourage out-of-model negotiation

Overlapping systems

- Incremental upgrades
- Redundancy

Information gathering

Expect repeated use

- Financial disclosure
- Surveys
- Audits

Costs

- Response time reduced by 70% over paper
- Software cost approaches $1/response
- Design, rollout costs are greater!

Early-adopter phase

- 1st-year savings of > 1 FTE *including startup*
- Training time < product lifetime

Participation

Online surveys get 50-80% response rates

- **Call to action**
- **Quid pro quo**
- **Participation in a meaningful decision**

Rewards

- **Attention is more powerful than a commodity**
- **Work with established relationships**

---

eForm technology

Data capture

- **Web browsers** *1-year*
- **Acrobat** *2-year*

Archiving

- **Custom programming** *4-year*

Reporting

- **Format** *forever*
- **Implementation** *4-year*

Digital signatures

- **7 of 10 Harvard graduates can’t figure out PGP**
- **Does a bitmap provide authentication?**

John Hancock

Software incompatibilities

Software lifetime is 1-2 years

How to make *Internet speed* work for you

Design for rapid replacement

Focus on cheap, creative improvements

Standards-based languages and formats

Avoid ’brand name’ inventions

Encourage human intervention

Fundamental laws

Hofstadter’s law *Any project involving computers will take longer than you think it will, even if you account for Hofstadter’s law.*

Moore’s law *Computing power doubles every 18 months.*

Your computer was obsolete when you bought it.

Today’s proprietary design is tomorrow’s

legacy code.

Every organization has unique needs.

Software will have unpredictable interactions.

XML

Works well for

- Form definition
- Documentation
- Reports
- Transactions (SOAP)
- Data structure

Works poorly for

- Data storage
- Word processing
- Programming language

Electronic Form System

Schema at electronicform.org and OASIS

Like HTML

Extra types *Text, password, number, date, time, radio, checkbox, weighted, rank-list, single list, multiple list, combobox,*

Data types designed around humans

Separate data structure from presentation

Good *flexible rearrangement*

Bad *not a word processor*

’Question’ paradigm

Good *abstraction from implementation*

(HTML, Palm)

Bad *requires sophisticated CGI*

Example

```
<form>

<fieldset>

  <text>Page 1</text>

  <input type=text name=Name size=30>

    <text>What is your name?</text>

  </input>

  <input type=listsingle name=Location>

    <text>Where do you live?</text>

    <option value=DC>Washington, DC</option>

    <option value=MD>Maryland</option>

    <option value=VA>Virginia</option>

  </input>

</fieldset>

</form>
```

Electronic Form System

Good

Searchable file system

Manual manipulation *Add/remove large groups*

- **Forward/backward compatibility**
- **Division of labor**
- **Database-agnostic**
- **Presentation-agnostic**
- **Encapsulation (form, report)** *version control, translation*

Bad

- **Inherently hierarchical** *"Q2 has the same responses as Q1, so when I change one, change the other."*
- **Limiting for certain new features**

Electronic Form System

Database independence

- **Scripting languages and JSDB middleware**
- **Use JS (Perl, Python) as glue for database, email, web**
- **Could write drivers for new formats (Notes, XML database)**

Presentation independence

- **Generate HTML, email, Palm**
- **Could write drivers for PDF, XForms, WAP**
- **Add new settings for new platforms**

Interactive XML editing

Obvious interface *tree plus property pane*

Use named references

Minimal information in tree (PowerPoint poisoning)

Value is not related to cost

Data model

Trash can instead of undo

Isolate display from XML!

Property sheets

GUI design expensive

Fast insertion of new features

Hide infrequently used properties

Encoding

Linefeeds? Embedded tags?

XML editing

Manual editing *no software development for one-time changes*

Collaborative editing

Documentation *generic editor*

Online help is stored a file with XML tags

Cut & paste HTML

Transform with JS (forgiving parser!)

Data model easily changed

Large XML files annoying

Forms *custom editor*

- Core information on main screen
- Pop-up ’advanced options’
- JS scripts for repetitive tasks

XML editing

GUI design

- What to put up front?
- How to treat relationships between tags?
- Property pages are annoying
- More like a file system than word processing or spreadsheet

The web is a poor interface

- Automatic refresh crucial
- Can’t debug
- Clients unreliable

Systems engineering

Fundamental tradeoff

- Complexity of interfaces
- Duplication of effort

What to do with duplicate information in an XML file?

- How many programs will interact with it?
- Expected lifetime?

XML format translation

Little Languages Law

- XSLT?
- JS, Perl for transforming formats

Custom languages

- Online reports, real-time calculation

Human interface

One file with all languages

- Database idiom
- Related information co-located
- Complex interface *hard to write programs*

Separate files for each language

- File idiom
- How many languages do you know anyway?
- Easier to write programs
- Duplication of effort

Example

```
<message>

<text language=en>Thank you</text>
```

```
<text language=ru>Spaseebah</text>

<text language=pt>Obrigado</text>

<text language=jp>Arigato</text>

</message>
```

This won’t work for so many reasons

Alternatives?

Drivers *Translate each time you re-generate the web site*

Better data model *Do you speak 4 languages?*

XML as a database

A. Edit file in memory, save on disk

Cheap to implement
Random read/write easy
No recovery from crashes or deletes
Requires two copies *memory, disk*

B. Changes written to a log file

More expensive
Read easy, write difficult
Reliable
Log file?

Complexity & robust control theory

Stability

We like clever algorithms *more reliable, less intervention*

Performance increases with complexity *Just-in-time delivery, Credit card theft detection, Derivatives trading*

Complexity improves handling of special cases

Robustness

How well do you handle the unexpected?

Complexity always decreases robustness

Well-meaning algorithms might make problems worse!

Tradeoff

The bigger they are, the harder they fall

You can never win

Complexity & robust control theory

Regulation

Gather information about system

Make change to system

Feedback *keep the pencil inventory steady*

Too few? Order more.

Too many? Transfer to another

stockroom.

Vicious cycle

1. Stockroom runs low on pencils
2. Computer orders more pencils
3. Delivery late with pencils
4. Computer orders more pencils
5. Too many pencils!

Complexity & robust control theory

Control system

Plant *Gather information about system*

Servo *Make change to system*

Disturbance *Other changes to system*

Servo + Plant *keep the measurement within some range*

What determines stability

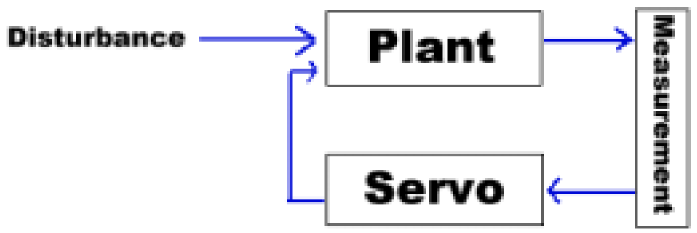
Time delays

Measurement errors

Construction errors

Power of servo to make changes

How rapidly do events occur?



Robustness

Expected problems with

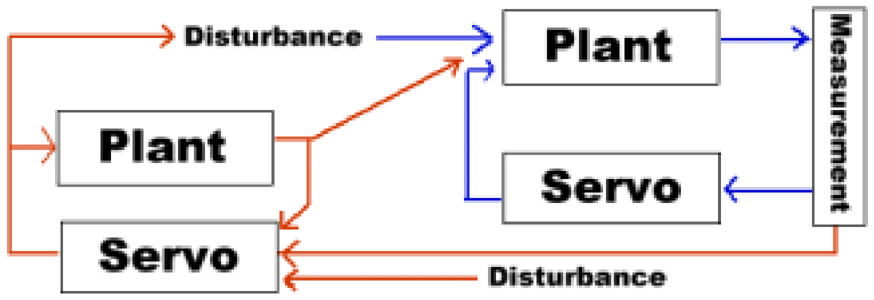
Fast disturbances

Large disturbances

Unexpected problems

Interactions with other servos

Escalating feedback



Prevent computers running amok?

Keep records of interactions

Who do I collect data from?

Who collects data from me?

Make records available to others

What risks am I taking by using this data?

Who will I affect by making this change?

Automatic processing

KISS

Make 'triggers' accessible

Simulation and oversight

You have been watching

Raosoft®

Shanti Rao

Raosoft, Inc.

www.raosoft.com

Electronic Form System

www.electronicform.org