Purpose

To translate BridgeNet XML into a display of the hands and results in HTML format.

Rough Process

1) Read xml file as a string into a variable
2) Use xmlminidom to process the string into "node"s
3) Iterate through all board nodes for each of these generate HTML
    1. Make boards with travelers
    2. Make boards without travelers
    3. Make personal score card
    4. Make rankings
4) Output html headers with style sheets, results from 3, and footers to files.

Data Structures

The only specialized data structures are for usage with xmlminidom:

```
(mv nodename attributes children) |  (mv 'text nil value)      ;node and text
attributes = (cons attribute attributes) | nil               ;attributes
attribute = (mv attributename value)                         ;attribute
nodes = (cons node nodes) | nil                              ;nodes
children = nodes                                             ;children
```

Interfaces and Contracts

xmlminidom:

xml-readnode (xmlchars) → returns the root node from xmlstring

Given any input xml-readnode will either return a structure of type xml-isnode or nil.

xml-getnodes (node nodename) → returns children of node with type nodename

xml-getdeepnodes (node nodename) → returns children of node with type nodename searching recursively using DFS with node as root.

Assuming (xml-isnode node) will return:

- something of type xml-isnodelist
- every node with name nodename.

xml-getnode (node nodename) → returns first child node with type nodename

xml-getdeepnode (node nodename) → returns first child node with type nodename searching recursively using DFS with node as root.

Assuming (xml-isnode node) will return:

- something of type xml-isnode
- every node with name nodename.

xml-getattribute (node attributename) → returns the value of node's attribute with name attributename

Assuming (xml-isnode node) will return a string (empty string if not found)

xml-gettext (node) → returns the composite of all text inside of a node

Assuming (xml-isnode node) will return a string (empty string if no 'text elements)

xml-isattribute (attribute) → returns true iff attribute is an mv of length 2 with both elements of the mv being strings

xml-isattributelist (attributes) → returns true iff attributes is nil or a list of mv's of length 2 with both elements of each mv being strings

xml-isnode (node) → returns true iff node is actually a node

xml-isnodelist (nodes) → returns true iff nodes is a list of nodes or nil

board:

serializeresults (xmlnodes) → serializes xmlnodes to HTML table with prefix and postfix appended before and after respecitvely

getseparateresults (xmlnodes) → serializes xmlnodes to a sequence of HTML tables corresponding to the seperate results for each player

serializeboards (xmlnodes) → Returns a serialization of the "board" class divs from xmlnodes, a list of xml nodes. The resulting serialization will look something like:
* *
* Board: 33 *
(33 comes from the contents of the BoardNo element)
* *
(N comes from the direction attribute of the Hand element)
* Dealer *
* Vulnerable *
*♠234
*
* *
* *
tablehead
<tr>
<td>A3</td>
(A3 comes from the SectionLabel + PairID-NS)
<td>A5</td>
(A5 comes from the SectionLabel + PairID-NS)
<td>5.0</td>
(5.0 comes from the contents of the TotalScore element)
<td> </td>
<td>120.0</td>
(120.0 comes from the contents of the MatchpointsNS element)
<td>120.0</td>

(120.0 comes from the contents of the MatchpointsEW element)
</tr>
**tabletail*)))*

serializehands (xmlnodes vulnerable dealer) → serialize hands into divs, where the class is the hand direction derived from xmlnodes, the "vulnerable" and "dealer" divs are added inside as necessary, and the cards are added to each hand via serializehandcards.
xmlnodes conforms to the minidom structure and is of type hand.
vulnerable is the text content of the Vulnerable node
dealer is the text content of the Dealer node

serializehandcards (xmlnodes) → serialize the hand cards from xmlnodes where xmlnodes is a list of node structures representing "Suit"s.
For...
<Suit symbol="S">832</Suit>
<Suit symbol="H">QT42</Suit>
<Suit symbol="D">A865</Suit>
<Suit symbol="C">A9</Suit>
You will get something like
*♠832
*
*♥QT42
*
*♦A865
*
*♣A9
*

psc:

getpsc (xmlnodes) → Given xmlnodes, this returns a string HTML table for each PSC
Notes: Recurses for every pairid, parsing out each pair's score card

getnameforid (pairid data) → The string pairid needs to define the direction (E-W). section (A, B), and Number (1,2,3) returns string list (list NameOfPerson1 NameOfPerson2)
Notes: Pulls a pair's names from the nodes data

getboardsforpair (pairid results) → Given string pairid, where PairID needs to define the direction (E-W). section (A, B), and Number (1,2,3), and node results returns a string Boards-HTML which is a HTML table with rows for each match
Board, Direction, Versus, Score, Matchpoints are the columns
Notes: Iterates over the list returned from getseperateresults to pull one pair's board results

rankings:

getrankings (rankings) → Given list of nodes rankings returns a string HTML will be some header information then a table for each Section/Direction pair
Columns: Pair No., Players, Strat, Overall Rank (A, B, C), Section Rank (A,B,C), Matchpoint Score, Percentage Score, Masterpoint Award
Notes: Parses out the rankings data into html tables

getcontestant (pairid rankings) → Given string pairid, where PairID needs to define the direction (E-W) section (A, B) & Number (1,2,3), and rankings nodes returns a list of Strings, one string value for each of the following:

Pair No., Players, Strat, Overall Rank (A, B, C), Section Rank (A,B,C), Matchpoint Score, Percentage Score, Masterpoint Award
Notes: Pulls a single pair's results out of the rankings data

io:

main(bridgeXML state)→Given a duplicate bridge XML file, extracts appropriate information to creates four HTML pages that link together: boards, boards with travelers, rankings, and personal score card webpages.

boards-no-trav(bridgeXML state)→Given a duplicate bridge XML file, gets the following information from each board listed in the XML file and creates an HTML page with this information formatted into tables:
Board numbers, dealer, vulnerable, and hands for each direction

boards-trav(bridgeXML state)→ Given a duplicate bridge XML file, gets the same information as boards-no-trav as well as the travelers information for each board, which includes the total score and matchpoints for all pairs at that board. Creates HTML page with this information with the board information and travelers information in separate tables.

rankings(bridgeXML state)→ Given a duplicate bridge XML file, creates a rankings table in an HTML file including each pairs ranking and various other stats such as matchpoint and percentage score.

personal-score-cards(bridgeXML state)→Given a duplicate bridge XML file, creates an HTML file containing personal score cards for each pair, which includes information about each match they played an against whom.

PROBE Software Size Estimate:

Reused Functions	*LOC*
xml-gettext	*9*
xml-getattribute	*8*
xml-getnode	*2*
xml-getnodes	*10*
xml-readnode	*5*
xml-readnodes	*53*
xml-skipdontcares	*20*
xml-readnodeproperties	*24*
xml-unescape	*27*
splitoff-prefix-mv	*3*
split-at-delimiter-mv	*3*
span-mv	*3*
split-on-token-mv	*3*
serializeboards	*32*
getresults	*35*
serializehands	*62*
serializehandcards	*32*
suit-list?	*8*
suit?	*38*

Total Reused LOC: 377

Historical Data Table for PROBE

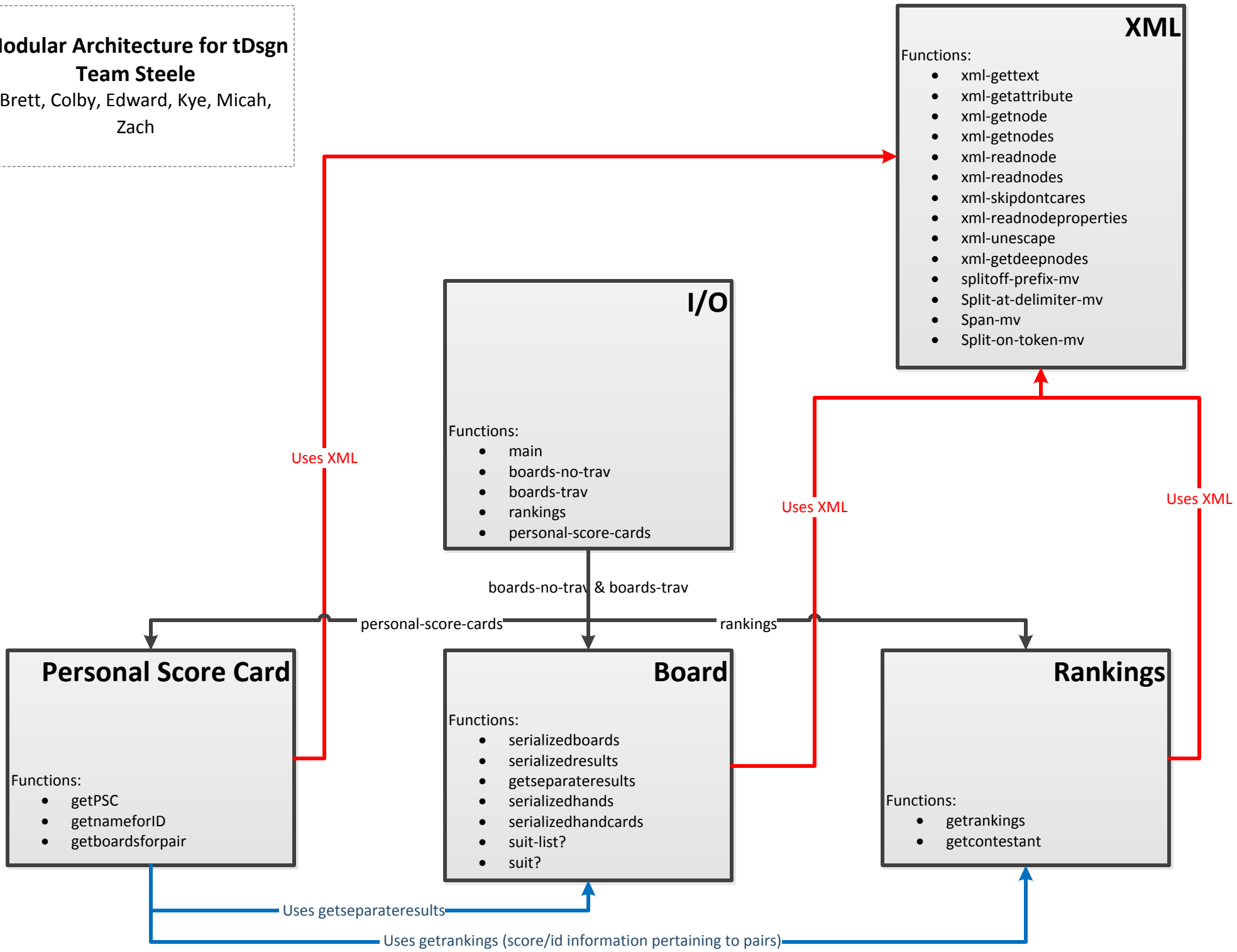
	Tiny(7%)	*Small(24%)*	*Medium(38%)*	*Large(24%)*	*Huge(7%)*
Avg. LOC	*1.69*	*3.82*	*7.34*	*16.31*	*41.71*

Predicted Functions	*Predicted Size*
xml-getdeepnodes	*Medium*
serializeseparateresults	*Large*
serializeresults	*Huge*
getPSC	*Large*
getnameforID	*Medium*
getboardsforpair	*Medium*
getrankings	*Medium*
getcontestant	*Medium*
boards-no-trav	*Small*
boards-trav	*Small*
rankings	*Small*
personal-score-cards	*Medium*
main	*Large*

Total Predicted LOC: 146.14

Total Estimated LOC: 523

Modular Architecture for tDsgn
Team Steele
Brett, Colby, Edward, Kye, Micah, Zach



tDsgn

Personal Software Process Summary

Project Essentials

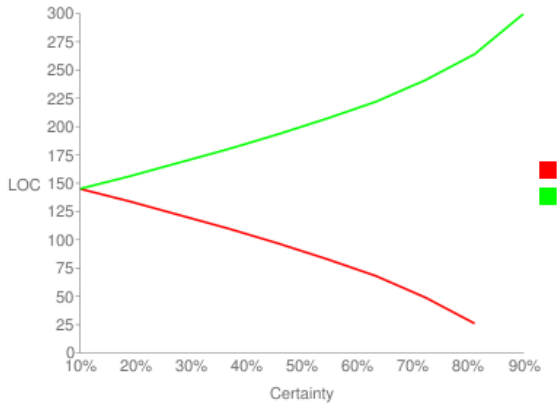
Name: Zach Bartlett
Instructor: Dr. Page
Date: Nov 11, 2010
Language: ---

Lines of Code

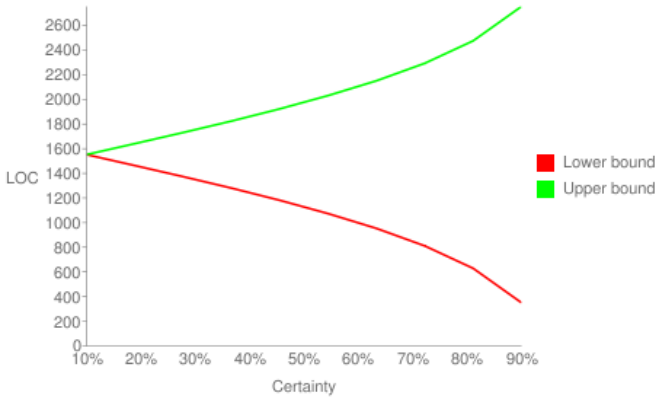
Type	Prediction by user	Actual
Added	102	0
Base	377	0
Modified	0	0
Removed	0	0

PSP Projection

LoC Certainty

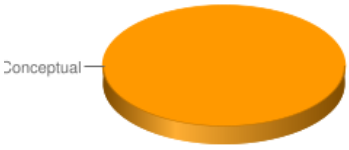


Time Certainty

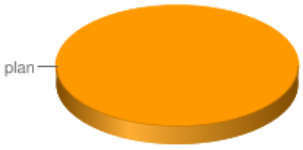


Project Data

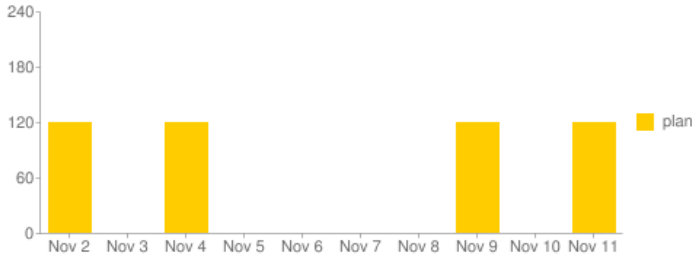
Time Per Defect Type

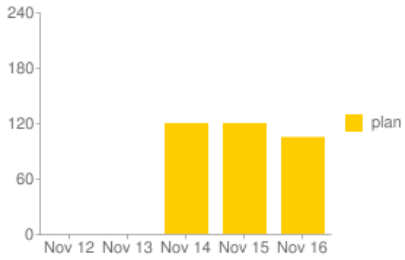


Time Per Phase



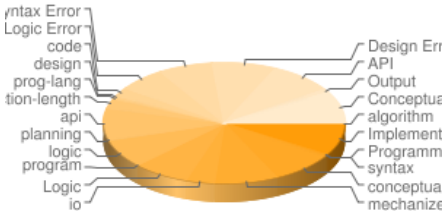
Time by Day



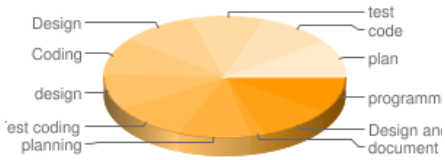


Cumulative Data

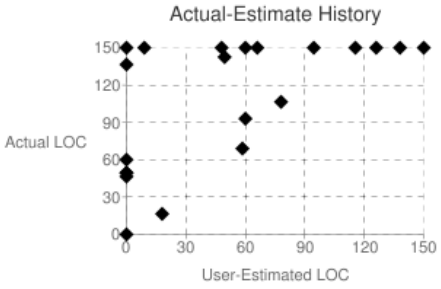
Time Per Defect Type



Time Per Phase



Actual vs Estimated LoC



Actual vs Estimated LoC

Project	Estimate	Actual
iEx5	40	62
Individual Exercise 5	77	156
Bridge Hands with Travelers (iEx5)	12	11
iEx5	33	95
iEx5	40	250
iEx4	84	143
Individual Exercise 4	112	101
Bridge Hand Displays (iEx4)	39	46
iEx4	32	244
iEx4	6	175
iEx3	92	145
Individual Exercise 3	52	71
Dealing Instructions from XML data (iEx3)	44	114
iEx3	0	290
iEx3	63	125
Individual Exercise 2	0	91
iEx2	0	0
Project 2	0	33

Individual Exercise 1	0	40
iEx1	0	31
Project 1	0	33

Time Log

Date	Type	Int. Time	Description
Nov 2, 2010, 3:00 PM - 5:00 PM	plan	0	Completed a rough sketch (design) of modules and interfaces. Discussed Project Description and requirements. Discussed previous code implementations to be used in tImpl.
Nov 4, 2010, 12:00 PM - 2:00 PM	plan	0	Worked more on the design, completed a visio diagram of modules and function definitions and assigned tasks to each team member for tDsgn. Looked more at the project description.
Nov 9, 2010, 3:00 PM - 5:00 PM	plan	0	Brought together the function descriptions, decided on github as our distributed version control system. Assigned additional assignments to team members. Looked even more at project description, noting possible difficulties.
Nov 11, 2010, 3:00 PM - 5:00 PM	plan	0	worked on PSP++ report, discussed the project description more to decide how to manage the linkage aspect. More difficult to do score cards and rankings. Set up github accounts, etc.
Nov 14, 2010, 6:00 PM - 8:00 PM	plan	0	Colby, Edward, and Zach worked on Design Review more. Renamed several functions in design to better explain what they should do. Expanded upon annotations of functions especially in psc / rankings.
Nov 15, 2010, 6:00 PM - 8:00 PM	plan	0	Kye, Brett, and Micah worked on Design Review. Updated PROBE data to be more accurate and appended IO module data.
Nov 16, 2010, 11:45 AM - 1:30 PM	plan	0	Group met to perform a final review of tDsgn and tDsgnreview. Looked over function contracts and descriptions, finalized psp data, finalized architecture.

Defect Log

Date	Phase	Fix Time	Description
Nov 4, 2010	Conceptual	15	Realized XML files given as examples had given schemas. Resulted in less work required to differentiate between placement of results tags for boards.
Nov 4, 2010	Conceptual	30	Wrote new xmlminidom (utilities) functions that performs a Depth First Search of nodes of a particular name.
Nov 9, 2010	Conceptual	20	Decided that getseparateresults needed a helper function to send results data that is not appended with html tags.

Nov 14, 2010	Conceptual	30	Decided that board functions required addition specification and new names (serialized...).