Plat_Forms 2007 Task: PbT

Lutz Prechelt
prechelt@inf.fu-berlin.de

Institut für Informatik
Freie Universität Berlin
Berlin, Germany
http://www.inf.fu-berlin.de/inst/agse

January 2007

Abstract

This document contains the requirements for the system to be built by the participants of the Plat_Forms 2007 contest. The system is called PbT (People by Temperament). It is to be written within 30 hours by a team of three people. For further details about the contest, please see www.plat-forms.org.

Contents

1 Introduction

1.1 Theme/purpose of the PbT system

PbT (People by Temperament) is a simple community portal where members can find others with whom they might like to get in contact: people register to become members, take a personality test, and then search for others based on criteria such as personality types, likes/dislikes etc. Members can then get in contact with one another if both choose to do so.

The system has both an interactive user interface via HTML pages and a WSDL/SOAP-based programmatic interface.

1.2 Requirements priorities and notation

The requirements described in this document are categorized into three different priority levels and each requirement is marked accordingly:

MUST
- MUST marks an essential requirement. Unless all of these requirements are implemented, the system is considered inacceptable.

SHOULD
- SHOULD marks an important requirement. If some of these requirements are not implemented, the system is considered incomplete, but acceptable.

MAY
- MAY marks an optional requirement. These requirements are considered nice-to-have but need not be implemented when time is short or if their cost-benefit ratio is considered too high.

Each requirement is marked by exactly one of these three terms, followed by a subscript number that is the unique reference number of that requirement (also repeated in the margin).

If a compound requirement (such as a use case) is marked MUST, that means at least the MUST requirements contained in it must be realized. It does not mean that all of its subrequirements must be realized.

If a compound requirement is marked SHOULD or MAY, it will be considered realized only if all of the MUST requirements contained in it are realized.

If you make tradeoffs among non-essential requirements, maximize the user value of the portal, rather than showing off your technical capabilities.

Phrases *in italics* in the use cases below are references to further use cases, the corresponding section number is appended in parentheses as a hyperlink.

2 Functional requirements: Use Cases

Each use case consists of three subsections: The mandatory main scenario section describes the standard sequence of events for the use case. The optional exceptions and variants section describes what happens in important error cases and which voluntary deviations from the main scenario should be considered. The optional notes section provides detail information where needed.

2.1 User actions overview

2.1.1 Main scenario

$MUST_1$ M 1

1. The *user registers* (2.2) to become a member.
2. The portal greets the new member with a few member statistics (MAY_2) and a list of things to do ($SHOULD_3$), in particular taking the temperament test. m 2 S 3
3. The *member takes the Temperament Test* (2.3).
4. The portal directs him to an explanation of the resulting type as posted on keirsey.com or mbti.com ($MUST_4$) and describes the list of possibilities for further action (MAY_5). M 4
5. The *member works with Search For Members* (2.4). m 5
6. The *member works with Status Page* (2.6).
7. The member logs out ($MUST_6$). M 6

2.1.2 Exceptions and variants

- 1b. The member logs in ($MUST_7$). M 7
  2b. The portal greets the existing member with the *Member List* (2.5) of members that have answered an RCD positively since the last view of the Status Page (MAY_8). m 8

2.1.3 Notes

- Request for Contact Details (RCD): From the point of view of member A, another member B can be in four different states: *no_contact* (the initial state) means no RCD has been sent between the two, *RCD_sent* means A has made an RCD to B, *RCD_received* means B has made an RCD to A, and *in_contact* means each has made an RCD to the other and both can see the other's contact details (full name and email address) which are normally hidden. Sending an RCD happens in the *no_contact* and *RCD_received* state only, the latter case is called "answering an RCD". A positive answer leads to the *in_contact* state, a negative answer to the *no_contact* state. RCD

2.2 User registers

2.2.1 Main scenario

M 9 MUST$_9$

Precondition: User is not logged in

1. The user enters the following mandatory information (see the notes):
   M 10 — full name, email address (MUST$_{10}$)
   M 11 — town, country (MUST$_{11}$)
   M 12 — life motto (MUST$_{12}$)
   M 13 — username, password (MUST$_{13}$)
2. The user enters the following optional information (see the notes):
   M 14 — secondary life motto (MUST$_{14}$),
   S 15 — list of likes (SHOULD$_{15}$),
   S 16 — list of dislikes (SHOULD$_{16}$),
   S 17 — GPS coordinates of place of residence (SHOULD$_{17}$),
   S 18 — primary Enneagram personality type (SHOULD$_{18}$),
   S 19 — secondary Enneagram personality type (SHOULD$_{19}$).
3. The user submits this data for registration.
4. The portal validates the username, registers the user as a new member, stores the data, and logs
   M 20 in the member (MUST$_{20}$).

2.2.2 Exceptions and variants

- 2b. The user can also add this information later using the Status Page.
- 4b. The portal rejects the username because it is not unique (already in use) and sends the user
  M 21 back to step 1 (MUST$_{21}$).
- m 22 4c. The portal MAY$_{22}$ validate the email address to have a valid domain.

2.2.3 Notes

- Steps 1 and 2 should be considered a whole and can be distributed over one or several dialog pages according to the chosen UI philosophy.
- Full name and email address are initially hidden from view for other members. Members can make them visible on a one-to-one basis by the RCD mechanism described in 2.5. All other data is considered public.
- The life motto is an arbitrary one-line phrase or sentence meant to characterize the person.
- The likes and dislikes are sets of words or short phrases that characterize things, activities, attributes, etc. that a person is fond of or cannot stand, respectively.
- m 23 The portal MAY$_{23}$ also offer to select from the union set of likes (or dislikes, respectively) sub-
  GPS mitted by all other members in order to create a more controlled vocabulary.
- M 24 GPS coordinate format: GPS coordinates are entered textually in decimal notation. They MUST$_{24}$ conform to the following regular expression: \d+[\.\d+]? ?[NnSs] ?,? ?\d+\[.\d+]? ?[EeWw]

- GPS coordinate precision: The user can freely choose the precision of the GPS coordinates (number of decimal places) and hence the precision with which the location is revealed.
- GPS coordinate determination: The portal MAY_{25} provide an explanation how to determine one's own GPS coordinates via Google Maps and its "URL for this page" link. m 25
- Enneagram type: This is a personality type. See http://www.9types.com for details. A type is a 1-of-n selection among "unknown", "1 (reformer)", "2 (helper)", "3 (motivator)", "4 (artist)", "5 (thinker)", "6 (loyalist)", "7 (enthusiast)", "8 (boss)", "9 (mediator)" and for the secondary type also "none" ($MUST_{26}$). The portal MAY_{27} offer hyperlinks with explanations of each type and $SHOULD_{28}$ guide the member to an Enneagram test on 9types.com for determining primary and secondary type. M 26 m 27 S 28

2.3 Member takes the Temperament Test

2.3.1 Main scenario

$MUST_{29}$ M 29

Precondition: User (member) is registered and logged in.

1. The portal presents the binary questions of the TTT (see notes).
2. The member answers the questions by selecting one of the two choices for each of the questions ($MUST_{30}$). M 30
3. The portal evaluates the answers, computes and stores the test result and displays the test result ($MUST_{31}$) along with the MBTI type ($MUST_{32}$), Keirsey temperament (MAY_{33}) and hyperlinks to explanations of each ($SHOULD_{34}$). M 31 M 32 m 33 S 34

2.3.2 Exceptions and variants

- 2b. The member may choose not to answer all of the questions. Individual questions can be left without an answer at will ($SHOULD_{35}$). S 35
- 3b. The portal rejects the answers because less than five answers have been received for at least one of the four dimensions ($SHOULD_{36}$, see notes). S 36

2.3.3 TTT, KTS, MBTI, E/I, S/N, T/F, J/P, type, temperament

- The TTT (Trivial Temperament Test) is a simple personality test along the lines of the Keirsey Temperament Sorter (KTS) as provided on keirsey.com. Its results have the same structure as those of the KTS. The TTT, however, was invented for the Plat_Forms contest only; it is coarse, unvalidated, and somewhat nerd-oriented and should hence not be taken too seriously. TTT
- You find the definition (questions, answer choices, and corresponding dimension indicators) of the TTT in the file `ttt-questions.txt`. The format is as follows: There is one 4-line block for each of the 40 questions of the TTT. Line 1 is the question, lines 2 and 3 are the answer choices (must be presented in this order) preceeded by the evaluation marker (indicating dimension and tendency) and a colon, line 4 is empty.

- The KTS classifies a person's personality along the four dimensions of the MBTI (Myers-Briggs Type Indicator), the world's most widely used personality assessment instrument. These dimensions are: E (extrovert) vs. I (introvert), S (sensing) vs. N (intuitive), T (thinking) vs. F (feeling), and J (judging) vs. P (perceiving); see Wikipedia and mbti.org for details. (Note that the discussion about what is similar or different between KTS and MBTI with respect to the types and dimensions is complicated and sometimes heated. We will consider them equivalent here. Also do not get distracted by the varying labels assigned to the two ends of each dimension; they do not matter for our purposes.) MBTI
- An MBTI personality type is a combination of four letters, one per dimension, e.g. INTP or ESTJ etc. (The dimensions are always given in this order.) There are hence 16 different types.
- A Keirsey temperament is a cluster of 4 MBTI types. There are hence 4 different Keirsey temperaments. They are: SJ ("Guardian"), SP ("Artisan"), NT ("Rational"), NF ("Idealist"). For instance, the ISTJ, ESTJ, ISFJ, and ESFJ types are all SJ ("Guardian") temperaments.
- TTT type is the same as MBTI type.
- The TTT works as follows:
  — Each question targets a certain tendency along one of the dimensions. For example, a particular question may refer to the E/I dimension. Then one of its answer choices indicates a tendency in the E direction, the other a tendency in the I direction.
  — Test evaluation just counts these tendencies across all questions and computes their difference within each dimension. For example if there are 10 questions for the E/I dimension, and the user chose the E answer 7 times and the I answer 3 times, the resulting type would contain E, and the full test result would indicate E+4.
  — Test results indicate how many more answers the stronger tendency in each direction had than the weaker one. For example E+4S+5T+1J+0 means we had four more E than I answers, 5 more S than N answers, 1 more T than F answers, and as many J as P answers. The resulting type in this case would be ESTJ, the temperament would be SJ.
  M 37 — In the case of ties (as with J vs. P above), the portal MUST_{37} always prefer I over E, S over N, T over F, and J over P. So we can have E+0, S+0, T+0, or J+0, but never I+0, N+0, F+0, or P+0.

2.4 Member works with Search For Members

2.4.1 Main scenario

M 38 MUST_{38}

Precondition: User (member) is registered and logged in.

1. The portal presents a search dialog with the following filtering choices (see notes for details):
2. Status-related choices:
   M 39 — only members who are not yet contacts of mine (MUST_{39})
   S 40 — only members who have not yet received an RCD from me (SHOULD_{40})
   m 41 — only members who have registered since my last logout (MAY_{41})
   m 42 — only members who have registered during the last 1/3/7/14/30/60/90 days (MAY_{42})
   m 43 — only members who have taken a TTT since my last logout (MAY_{43})
   m 44 — only members who have taken a TTT during the last 1/3/7/14/30/60/90 days (MAY_{44})

3. Location-related choices:
   — only members in my country (MAY_{45}) m 45
   — only members who live less than 5/10/20/50/100/200/500/1000/2000/5000 kilometers away (SHOULD_{46}) S 46
4. Personality-type-related choices:
   — only members with the following TTT types <types> (MUST_{47}) M 47
   — only members with the same primary Enneagram type (MAY_{48}) m 48
   — only members with one of my Enneagram types as one of theirs (MAY_{49}) m 49
   — only members with one of their Enneagram types related to one of mine (MAY_{50}) m 50
5. Like/dislike-related choices:
   — only members whose life motto (any one) contains <string> (MUST_{51}) M 51
   — only members who share one of my likes (SHOULD_{52}, and MAY_{53} be extended into ”share at least 1/2/3 of my likes”) S 52 m 53
   — only members who share one of my dislikes (SHOULD_{54}, and MAY_{55} be extended into ”share at least 1/2/3 of my dislikes”) S 54 m 55
   — only members who dislike none of my likes (SHOULD_{56}, and MAY_{57} be extended into ”dislike at most 0/1/2 of my likes”) S 56 m 57
   — only members who like none of my dislikes (SHOULD_{58}, and MAY_{59} be extended into ”like at most 0/1/2 of my dislikes”) S 58 m 59
6. The member selects some choices (at least one) and submits the search
7. The portal finds all members that satisfy all of the filters and presents them as a *Member List* (2.5) that the member can then work with.

2.4.2 Notes

- x kilometers away: The comparison is performed based on the GPS coordinates alone. You need not implement spherical geometry computations, rather you MUST_{60} use the following simple substitute: with a longitude difference of x and a latitude difference of y, the distance d is $d = \sqrt{x^2 + y^2}$, where the unit of d is 100 kilometers. If you make sure you choose the shortest way, this is correct with respect to latitude differences and tends to overestimate longitude differences (except near the equator, where it underestimates, since $40,000/360 \approx 111$). GPS M 60
- GPS data precision: The low precision that some users may have chosen for their GPS coordinates is ignored in the distance computation. All data is treated as if it was arbitrarily precise.
- You SHOULD_{61} devise a nice interaction style for specifying a set of types. It needs not be capable of representing arbitrary sets; wildcards (”don’t care”) for individual dimensions are sufficient. It is useful if the member can quickly recall selections used in previous searches (MAY_{62}). S 61 m 62
- Related Enneagram type: Enneagram theory talks about relationships from each type to two other types, marked by arrows to and from in Figure 2.1. For example for somebody with type 1, the related types are 4 and 7. For somebody with types 1 and 2 they would be 4, 7, and 8.
- Choices that make no sense because of missing data (e.g. lack of TTT result, Enneagram type, or dislikes list for the current user) SHOULD_{63} be disabled. S 63

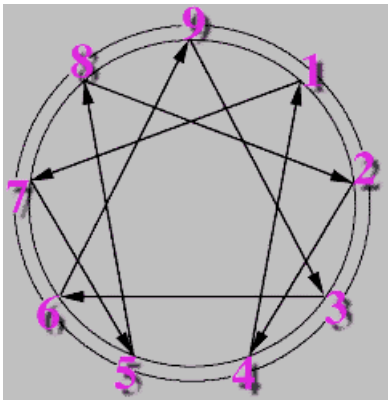

Figure 2.1: Enneagram type relationships

2.5 Member works with Member List

2.5.1 Main scenario

M 64 MUST$_{64}$

Precondition: User (member) is registered and logged in.

1. The portal displays a graphic with a 2-dimensional overview plot of the distribution of the members in the list (MUST$_{65}$, see notes). M 65
2. The member changes which variables are plotted in the graphic (MUST$_{66}$). M 66
3. The portal displays an updated graphic.
4. The portal shows the members of the list with the following attributes: username, town, country, life mottos, TTT type, and Enneagram types (MUST$_{67}$). M 67
5. The member selects some members whom s/he has not yet sent a Request For Contact Details (RCD, MUST$_{68}$, see 2.1.3). M 68
6. The portal sends an RCD to these members (MUST$_{69}$). M 69
7. The member requests to see the Status Page of one member from the list (MUST$_{70}$). M 70
8. The portal *shows the Status Page* (2.6).

2.5.2 Exceptions and variants

- 1b, 4b: If the member list is empty, the graphic is suppressed and the list replaced by a message (MAY$_{71}$). m 71

2.5.3 Notes on the graphical plot

Plot: Each member in the list (plus the user) is shown by one symbol positioned along two axes. See Figure 2.2 for an example. The horizontal and vertical axis each represent one of the following values for the member:

- E-I (positive values x for test results containing E+x, negative ones for I+x, MUST$_{72}$), M 72
- S-N accordingly (MUST$_{73}$), M 73

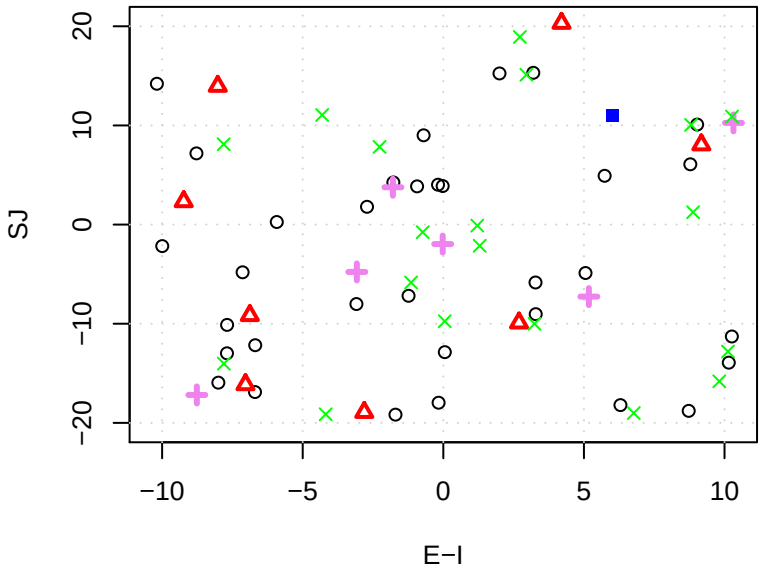

Figure 2.2: Member List overview plot (example)

M 74
- T-F accordingly (MUST$_{74}$),

M 75
- J-P accordingly (MUST$_{75}$),
- SJ (the sum of S-N and J-P, MAY$_{76}$), m 76
- SP (the sum of S-N and P-J, MAY$_{77}$), m 77
- NT (the sum of N-S and T-F, MAY$_{78}$), m 78
- NF (the sum of N-S and T-F, MAY$_{79}$), m 79
- distance-to-member (based on GPS coordinates, see 2.4.2, SHOULD$_{80}$), S 80
- like-overlap (number of shared likes, plus number of shared dislikes, minus number of disliked likes, minus number of liked dislikes, MAY$_{81}$) m 81
- membership-age (time since registration, MAY$_{82}$) m 82
- TTT-age (time since the TTT-taking that determined the current TTT type, MAY$_{83}$). m 83

Plot scattering: For the integer-valued data, the actual values are randomly scattered around in the plot by plus/minus 0.33 so that symbols are less likely to cover up one another (MUST$_{84}$). Use a uniform distribution for the scattering M 84

Plot colors: Discriminate the members shown by their relationship status (see 2.1.3) via different symbols (MAY$_{85}$) and colors (MUST$_{86}$, blue for self, black for no_contact, red for RCD_sent, green for in_contact, and violet for RCD_received). m 85 M 86

Plot details: As with the rest of the user interface, the details of the plot are free for you to arrange as appropriate. The example figure is illustrative, not prescriptive.

Plot interaction: Clicking on the symbol of a member in the graphic MAY$_{87}$ call that member's *Status Page* (2.6) m 87

2.5.4 Other notes

- Sorting: The list MAY$_{88}$ allow sorting by each attribute. m 88
- Long lists (over 50 members) SHOULD$_{89}$ be paginated. S 89

2.6 Member works with Status Page

2.6.1 Main scenario

M 90 MUST$_{90}$

Precondition: The user is logged in as member A

1. The Portal presents a status page about the member A that contains the following information:
M 91 — I: the set of information that is usually submitted on registration (MUST$_{91}$, see 2.2)
— M: a link to an external Google Maps page that will show a map (about 100 km across) of the
m 92 area around the member's given GPS coordinates (MAY$_{92}$)
M 93 — T: the result and timestamp of the last TTT-taking (MUST$_{93}$)
M 94 — C: the *Member List* (2.5) of members with in_contact status (MUST$_{94}$)
S 95 — S: the *Member List* (2.5) of members with RCD_sent status (SHOULD$_{95}$)
M 96 — R: the *Member List* (2.5) of members with RCD_received status (MUST$_{96}$)
S 97 2. The member reviews and modifies some or all of the information I (SHOULD$_{97}$).
M 98 3. The member selects some members from the RCD_received list and answers positively (MUST$_{98}$).
M 99 4. The portal updates the RCD_sent and in_contact lists (MUST$_{99}$).
M 100 5. The member selects some members from the RCD_received list and answers negatively (MUST$_{100}$).
M 101 6. The portal updates the RCD_sent list (MUST$_{101}$).
M 102 7. The member calls the Status Page of a member X from any of the lists (MUST$_{102}$).
8. The portal presents the *Status Page* (2.6) of X.

2.6.2 Exceptions and variants

- If the Status Page shown is not about member A, but rather about a different member B, the
M 103 member lists S and R are not shown and steps 2, 3, 4, 5, 6 are not possible (MUST$_{103}$).
- Unless B has in_contact status, the contact details (full name, email address) are also not shown
M 104 (MUST$_{104}$).
- 2b: The member takes another TTT. The portal stores the new result and uses it in place of the
S 105 previous one (SHOULD$_{105}$).
m 106 - 2bb: The portal MAY$_{106}$ store not just one, but rather several (or even all) previous results of TTT-takings. The member can list all previous results (with timestamps, in chronological order) and select one of them to be used as his or her "official" result which the other members get to see.

2.6.3 Notes

S 107 - Long member lists SHOULD$_{107}$ be paginated.
m 108 - The Status Page MAY$_{108}$ actually be realized as multiple separate pages.
- Google Maps: Information about how to construct a suitable Google Maps URL can be found on the web (but not at Google). The most relevant parameters are `z` and `ll`.

3 Functional requirements: Web Service interface

3.1 General information

- The web service interface consists of a single port type PbtServices and a single SOAP RPC binding for it.
- Its WSDL specification is available in the file `PbT.wsdl`.
- Your implementation of the service MUST_{109} be bound to URL `/soap` on your server. M 109
- The WSDL of your final implementation MUST_{110} be bound to URL `/wsdl` on your server. M 110
- With a single exception (clearDatabase, Section 3.3), the operations in the service follow the use cases described in Section 2 quite closely and represent most (but not all) of their underlying business functionality.
- The service performs no elaborate exception handling. Rather, illegal calls or calls to operations you have not implemented MUST_{111} simply return a nil result. M 111
- All of the WSDL specification MUST_{112} be implemented, but only those parts of the underlying functionality actually need to be present that you have also implemented on the usecase level (that is, in the HTML user interface). Therefore, input parameters that represent functionality you have not implemented on the usecase level MUST_{113} simply be ignored and output attributes that represent functionality you have not implemented on the usecase level MUST_{114} simply be returned as nil. The one exception of this rule is the clearDatabase operation, which must be implenented in any case. M 112 M 113 M 114
- Beyond the elementary types and arrays thereof, there is only one single complex type (plus arrays thereof) that is used in the operations: Memberinfo represents the details about one member roughly as represented on the status page. See the WSDL for details.

The subsequent descriptions of the individual operations can only be fully understood in conjunction with `PbT.wsdl` (because only parts of the signatures are discussed here) and the usecases in Section 2 (because those describe most of the semantics).

3.2 complexType Memberinfo

The attributes of `Memberinfo` represent the data about a member that is (or may be) shown to other members in a member list or on a status page.

- The `rcdStatus` is one of the strings ”no_contact”, ”RCD_sent”, ”RCD_received”, or ”in_contact” and MUST_{115} be expressed from the point of view of the member that has requested the Memberinfo. A member has ”in_contact” status relative to him/herself. M 115

- The contact details `fullname` and `emailAddress` will be nil unless `rcdStatus` is ”in_contact”
- `gpsCoordinates` follows the format convention described in Section 2.2.3
- All other attributes should be self-explanatory.

3.3 operation clearDatabase

`clearDatabase` resets PbTs complete internal state to what it was just after the initial deployment:
M 116 no users are registered, no current sessions or TTT results or RCDs exist ($MUST_{116}$).

This operation is needed to support the evaluation of the system, in particular load testing.

3.4 operations submitMemberinfo, getMemberinfo

M 117 `submitMemberinfo` is used for registration ($MUST_{117}$, in this case `sessionId` must be nil and `username` must not be previously used) and is also when a member modifies his/her information
M 118 via the status page ($MUST_{118}$, in this case `sessionId` must be a valid id as obtained by operation `login` and `username` must be nil).

All other parameters should be self-explanatory after reviewing the corresponding usecases in Sections 2.2 and 2.6.

M 119 `getMemberinfo` ($MUST_{119}$) obtains a `Memberinfo` object that contains the information stored by `submitMemberinfo` as described in Section .

3.5 operations login, logout

M 120 `login` ($MUST_{120}$) authenticates a member via `username` and `password`. If the authentication fails, the call returns nil. Otherwise, it returns a `sessionId` string that is used for identification/authentication of the *current user* in subsequent calls of other operations.

M 121 `logout` ($MUST_{121}$) invalidates a `sessionId` previously created by `login`.

3.6 operation takeTtt

M 122 `takeTtt` ($MUST_{122}$) realizes the core of usecase 2.3. It evaluates one set of answers to the TTT, computes the TTT result and TTT type, and stores them (plus a timestamp) for the current user.

`answers` is a string that contains one character for each question in the TTT (although not necessarily in that order). The character represents the answer given for that question. Assume character x represents the answer to a T/F question, then x is either T or F or blank; likewise for the other kinds of questions. Blank means the question has not been answered and will not be counted.

3.7 operations searchForMembers, searchForMembersGraphic

M 123 `searchForMembers` ($MUST_{123}$) realizes the core of the usecase 2.4. It takes many parameters describing the search filters and returns an array of `Memberinfo` objects that represent the members thus found as described in Section 3.2.

Unused filters are indicated by nil for string, false for boolean, 0 for int where 0 is a null-filter, 999999 for int where infinity is a null-filter. tttTypes is an array of TTT type strings (such as ”ESTP” etc.) that indicates the set of types of interest.

`searchForMembersGraphic` ($MUST_{124}$) performs the same search, but instead of returning `Memberinfo` objects, it returns the byte sequence of a GIF or PNG image file as described in Section 2.5.3. The image is `xSize` pixels wide, `ySize` pixels high and indicates the values of `xVariable` along the horizontal axis and `yVariable` along the vertical axis; the variables are given by their names as described in Section 2.5.3. M 124

3.8 operations getMemberlist, getMemberlistGraphic

`getMemberlist` ($MUST_{125}$) realizes the core of the usecase 2.5. It retrieves the `Memberinfo` of all members that have the given `rcdStatus` from the point of view of member `username`. M 125

`getMemberlistGraphic` ($MUST_{126}$) is to `getMemberlist` what `searchForMembersGraphic` is to `searchForMembers`. M 126

For both operations, if the current user is not allowed to get the list (because `username` is not the current user and `rcdStatus` is not ”in_contact” or because `rcdStatus` is ”no_contact”), nil is returned.

3.9 operation sendRcd

`sendRcd` ($MUST_{127}$) realizes an aspect of usecases 2.5 and 2.6: it sends an RCD from the current user to the members indicated in the `username` array or answers an RCD received from these members. Sending and positive answering is exactly the same: `positive` is true. Negative answering happens when `positive` is false. Negative answering to a member that had not send an RCD has no consequence whatsoever. Sending an RCD to a member with in_contact status has no consequence whatsoever. M 127

4 Non-functional requirements

4.1 User interface

M 128 The user interface MUST$_{128}$ conform to the above-mentioned requirements (as far as they are realized at all) in a sensible way with respect to the arrangement and markup of its elements and the labels, prompts and explanations that guide the user. Within those limits, the organization and design of the interface is left to the professional judgement of the participants.

S 129 The userinterface SHOULD$_{129}$ provide sufficient explanation of all uncommon concepts to guide a user who does not have prior knowledge about these topics. Make use of external links where needed. Do not include copyrighted material without permission.

m 130 Elaborate graphical design (MAY$_{130}$) is not important, but CSS SHOULD$_{131}$ be used throughout to
S 131 simplify future improvements.

m 132 It would be nice if the user interface works even when Javascript is turned off in a user's browser (MAY$_{132}$).

4.2 Browser compatibility

M 133 • The portal MUST$_{133}$ work fully with Firefox 1.5 and higher.

M 134 • The portal MUST$_{134}$ work fully with Internet Explorer 6 and higher.

S 135 • The portal SHOULD$_{135}$ work fully with Safari 2 and higher.

S 136 • The portal SHOULD$_{136}$ work fully with Opera 8 and higher.

m 137 • The portal MAY$_{137}$ work fully with other browsers such as Konqueror, Opera Mini, Lynx etc.

S 138 • If the portal relies on Javascript in the browser and Javascript is unavailable, it SHOULD$_{138}$ either produce a clear message saying that the portal is not functional (and why) or fall back to reduced (but still useful) functionality.

4.3 Scalability

M 139 The system MUST$_{139}$ scale without problems to 1,000,000 registered members.

M 140 For a realistic mix of requests, it MUST$_{140}$ have response times below three seconds (for at least 90% of the requests) up to at least 200 concurrently active users on a server with 1 CPU, 1 disk and 1 GB of RAM.

S 141 It SHOULD$_{141}$ have such response times up to at least 1000 concurrent users.

4.4 Persistence

All registration, temperament test, and RCD status information MUST_{142} be stored in persistent storage and must survive system shutdowns and crashes intact. M 142

4.5 Session timeout

Sessions MUST_{143} time out after one hour, both at the UI level and the webservice level. M 143

4.6 Integration

No integration with other systems is required. Where calls to other services are required, they are performed by the user's browser by following a URL supplied by the system. Where calls from other systems are required, they are the other system's responsibility and have to be performed via the web service interface.

4.7 Programming style

All identifiers and comments in the source code and helper files MUST_{144} be in English. M 144

Each source code file MUST_{145} be documented at least globally (purpose, called by). M 145

Each non-trivial public program element (such as a method) MAY_{146} be documented (purpose, usage). m 146

5 Rules for development

5.1 What is allowed

During the contest you may:

- Use any language, tool, middleware, library, framework, and other software you find helpful.
- Reuse any piece of any pre-existing application or any other helpful information you have yourself or can find on the web yourself. Anything that already existed the day before the contest started is acceptable.
- Use any development process you deem useful.
- Ask the organizer (who is acting like a customer) any question you like regarding the requirements and priorities (see 5.4).

5.2 What is not allowed

During the contest you may not:

- Disturb other teams in their work.
- Send contest-related email to people not on your team or transfer the requirements description (or parts thereof) to people not on your team.
- Have people from outside of your team help you. (This includes reusing work products from other teams.) There are two exceptions to this rule: (1) you may use answers of the customer as described in Section 5.4and (2) you may use user-level preview feedback as described in Section 5.3.

5.3 User feedback: The blog

m 147 During the contest, teams MAY_{147} showcase intermediate versions of their PbT service to obtain user-level comments and feedback. To do this, host your PbT service on your development server, open it for public access, and post a notification in the Live Contest Blog as described below.

Users can then comment on your PbT prototype regarding functionality, defects, usability etc. The teams are allowed to use this user-level feedback for improving their system. They are not allowed at this stage to post source code or to use information from outsiders that is on the code level.

5.3.1 How to post a message in the blog

1. Log in on plat-forms.org. Your username (team1, team2 etc.) and password (a 5-digit number) for the Blog is written on your CD or is available from Lutz Prechelt.
2. Go to 2007, go to Live Contest Blog
3. Rightmost column: Click ”New entry” in the Blog Manager
4. In the ”Edit” form, fill the ”title” line. Mention the names of your home organization, platform, and the version number of your new prototype, say, ”O’Reilly Perl team: Version 6”.
5. Fill the ”text” section. Indicate for which aspects you are particularly interested to receive feedback and perhaps what people should expect from your prototype. Most importantly, provide the URL where to access your prototype. This text should be a well-formed HTML fragment.
6. The ”text” section should be short. If you want to provide lengthy explanations, please put them in the ”more” section. In the blog overview, they will then be hidden behind a ”More...” link.
7. Further down, click the Category ”Live Announcement”
8. Store your data by clicking ”Save”. Note that your entry is *not* yet visible!
9. Choose ”Publish” from the ”State:” pulldown menu in the green title bar
10. If necessary, you can still modify your entry by navigating to it and performing ”retract” in the ”State” menu and then ”Edit”, both in the green title bar. Afterwards ”Save” and ”Publish” as before. Alternatively, you can delete it altogether clicking the red X on the right of the entry title.
11. While your entry is published, all visitors of www.plat-forms.org can leave a comment on the entry. As the author of the entry, you can (and should) delete comments you consider spam or trolling by clicking the red X to the right of the comment.

5.4 Talking to the customer

The customer of the PbT project is represented by Lutz Prechelt. He will be more or less available for questions regarding the requirements during most of the day and evening. He will not be available between midnight and 9:00 in the morning.

He will happily answer questions regarding clarification of the meaning of requirements (but beware: if his advice and this document should ever be in conflict, it is this document that is relevant for the evaluation, not his advice). He will only vaguely answer questions regarding requirements priorities or regarding which of two concrete solution ideas he would prefer and will point you to this document instead. He will not look at your solutions or give concrete feedback about them.

5.5 Delivery

You are allowed to finish your development at any time you think appropriate, but no later than the end of the 30-hour contest time. Early delivery will be taken into account during the evaluation.

Package and submit your deliverables as follows:

1. Shut down the virtual machine that is running your PbT service and put all files that make up the virtual machine into a ZIP file called `vmware.zip`. The virtual machine MUST$_{148}$ be set M 148

up in such a way that your PbT automatically starts up when the machine is booted and that your PbT does not need manual shutdown when the machine is shutdown. (The machine will always get at least 10 seconds of idle time before shutdown.)

2. Package a snapshort of your versioning database into a ZIP file. The versioning database is the subtree in file system of your versioning server that contains the directories and files holding all versions of each file of your PbT project you have under version control. If you used CVS as your versioning system, call the ZIP file `cvs.zip`; if you used Subversion, call the ZIP file `svn.zip`; etc.

3. Create a source code distribution of your PbT implementation and package it in a ZIP file called `pbt-sources.zip`. This distribution should contain all files needed to recreate an instance of your PbT service except those that already existed before the contest and were not modified. So you should include all source code, build files, skripts, configuration files, documentation, etc, if they are new or were modified. You need not include pre-existing infrastructure such as application server, database server, compiler, libraries, unmodified parts of frameworks etc. The content of this ZIP file wil be considered published under the OpenSource license(s) you specified when you requested participation in the contest.

4. Create a ZIP file named like <homeorganization>-<platformname>.zip, e.g. Optaros-perl.zip.
M 149 This ZIP file contains the three other ZIP files mentioned above and is your deliverable (MUST_{149}).

5. Determine the 160-bit SHA-1 message digest checksum of the deliverable. We will call this checksum the ”fingerprint” of the deliverable. SHA-1 is computed for instance by the sha1sum utility from GNU coreutils.

6. Send a two-line email containing the name of the deliverable and the fingerprint to all of the following email addresses: prechelt@inf.fu-berlin.de, he@heise.de, js@heise.de, richard.seibt@gmx.de.
M 150 The reception time timestamp of this email will be your submission time and MUST_{150} be before the end of the contest. Use the name of the deliverable as the subject of the email. The organizers will ignore all but the last such email from each team.

7. Fill in the 12 questions in the file postmortem-questionnaire.rtf. We need a separate questionnaire from each member of your team. Since you may not have enough energy left to provide sensible answers today, you need not do this right now; it is OK if you send the filled-in questionnaire at any time until Friday next week (2007-02-02). However, the information you give us in the questionnaire is important input for the scientific evaluation, so we kindly ask that each team member thoroughly provides his/her own personal answers. Please send the result to prechelt@inf.fu-berlin.de. Thanks!

8. Burn a DVD containing the deliverable. Write the name of the deliverable and the time of day on it with a pen. Hand the DVD over to the customer (Lutz Prechelt). YOU ARE DONE. Go and sleep or party or bang your head against a soft wall — whatever you feel most like doing. Thank you veeery much for participating in Plat_Forms!

M 151 **The fingerprint sent in your email absolutely** MUST_{151} **match that of the ZIP file on the DVD or else your whole participation was in vain.** The fingerprint email has two purposes: (1) to make the submission timestamp independent of your DVD burning.progress and (2) to validate the DVD, in particular a replacement DVD should the original one turn out to be unreadable.