MAINTAINING AND MEASURING MENTAL WELLNESS M3W	**M3W • Maintaining and Measuring Mental Wellness**	14/07/2015

Project	
Reference:	**AAL-2009-2-109**
Short Name:	**M3W**
Full Name:	**Maintaining and Measuring Mental Wellness**
Website:	http://m3w-project.eu

D16 - Data Processing and Evaluation

Document			
WG / Task:	WG1/6T	**Deliverable number:**	D16
Issued by partner:		**Confidentiality status:**	Public
Due date:		**Acceptance date:**	31/072015
Document status:		**Pages:**	40

Authors	**Name**	**Organization/Unit**
	Béla Pataki	BME-MIT
	László Ketskeméty	BME

Document History

Date	**Affected**	**Description of change**	**Author**	**Status**
2014/15		notes launched and spread	B. Sick	
15/07/2015		perusal	P. Breuer	Pre-final
31/08/2015		Formatting	P. Hanák	Final

Approval	**Name**	**Organization/Unit**

Content

1 Problems to be solved by data processing

The basic conceptual architecture of the proposed system is shown in Figure 1. The final goal is to provide appropriate long-term feedback to the user (or to the caregiver, family member, medical expert, etc.). Short-term feedback is for motivation to continue participating in the monitoring (“Well done!”, “Play some more games!”). Long-term feedback is the result of the change detection estimation: whether a significant change of mental state has occurred or not.

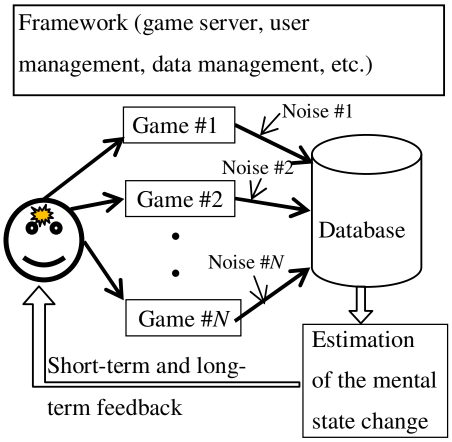

Framework (game server, user management, data management, etc.)
Game #1
Noise #1
Noise #2
Game #2
Database
Noise #N
Game #N
Estimation of the mental state change
Short-term and long-term feedback

Figure 1 Basic conceptual model of the cognitive state estimation system

Beyond the general problems of such systems (e.g., data privacy concerns), this approach has its special challenges, some of these are given:

1) How to measure the ***cognitive performance*** using computer games?
2) How to cope with the sometimes heavy ***noise*** of the uncontrolled (home) measurement environment?
3) How to ***motivate*** people to take part in the long run?
4) How to compare performance shown in different games, which is basically a special ***sensor-fusion*** problem?

1.1 Basic considerations

To ***measure the cognitive performance*** three principles are followed:

- To ensure the opportunity of measurements, proper serious games are selected, special ones are developed or clinical tests are modified taking into account the special requirements. Usually, games are modified to improve measurement capability; and tests are modified to be more entertaining. Most of them are logical puzzles, or they need the intensive use of the short-term memory (which is one of the best indicators of MCI), but other important parameters (attention, execution, language skills, etc.) are targeted as well. Two basic parameters are measured: the solving time of the puzzle and the good/bad steps taken during the solution. Currently only successful solutions are measured, for future work there are possibilities in the evaluation of the failed ones as well.

- Because the measurement of the mental state on an absolute scale is very hard, only the change in the person's performance is to be detected. For measuring a change, a reference is needed. There are two possibilities: the performance could be compared to a reference group; or it could be compared to a previously measured reference of the same person. Because the inter-personal comparison is affected by several parameters unknown in this voluntary, uncontrolled method (education, physical abilities, family conditions, profession, environment, etc.) the comparison in time to his/her own previous performance was chosen. However, since many people like to compare their own abilities to others' and to compete with others, such functionalities are offered as well.

- Because the voluntary measurement using computer games is very noisy (see section 1.2) evaluation is not performed in a 1-game basis. Only sets of several game sessions are compared to each other.

1.2 Noise and outliers

According to our experiments, the ***noise*** can be modeled using two terms:

- zero mean low level noise caused by the random differences between the consecutive puzzles and by minor environmental disturbances,
- major disturbances causing outliers.

The first term is eliminated by the averaging effect of the evaluation method (see later): several game results are evaluated together. The second term is an impulse like noise caused by the physiological, environmental and social disturbances resulting in outliers (for example, the telephone is ringing; the person has to use the bathroom, a storm is arriving, neighbor is coming, etc.). This second problem is solved by a filtering step, the outliers are simply rejected; they are not used in further evaluation steps.

These two types of noises are shown in Figure 2 for playtimes of a given player. That user solved nearly 120 puzzles during some months. There are natural fluctuations and one outlier. Naturally the less is the noise; the better is the measurement power of the parameter. As detailed in the next section, unfortunately the noise cannot be suppressed totally in the measurement. Users require some noise like fluctuation to maintain the entertainment of the game.

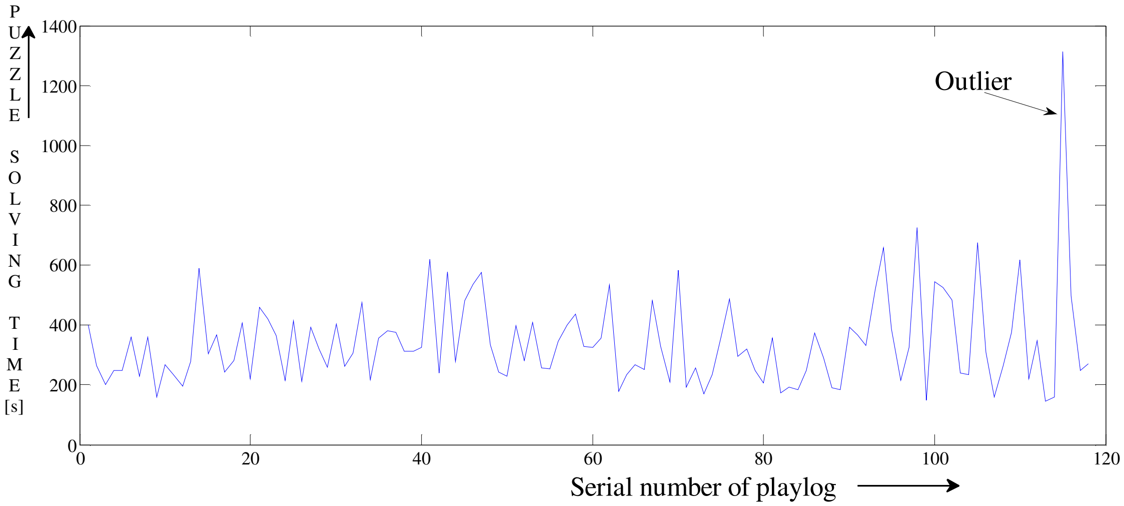


Figure 2 Typical time series of puzzle solving times (playtime)

1.3 *Entertainment capability and measurement power*

Early detection is the purpose; but the main problem is that nobody knows when the abnormal change will happen; maybe in some persons' life never. Therefore, the motivation must be managed probably for many years. It is a very complex problem itself; only some aspects are discussed here. Among several other aspects, one basic assumption is that although there is an extrinsic motivation that everybody wants to sustain mental abilities and an independent life of good quality, but generally it is not enough in the long-run. There must be intrinsic motivations too, e.g., entertaining ways of measurement, and short-term feedback (Figure 1) given to the user to encourage further playing (e.g., scoring or encouraging messages such as "Well done!" could generate motivation). An interesting example of the motivation thirst that scoring of the game sessions was not planned at first, but several players fed back their demand for it.

Unfortunately the entertainment capability contradicts the measurement power of the game in most of the cases. First problem is that if a game produces the same results for a person having a stable cognitive state (e.g. each time 600 points score) it is boring. At least some random behavior is needed to be entertaining, therefore some noise will be present.

In Figure 3 the total playtime played by all the players and some similar parameters are shown for the 15 most popular games. The games were implemented and made available at different times. Therefore, the total playtime is less informative than the playtime divided by the number of days when the game was available. The games are sorted according to that parameter. The parameters of the "best" game (Labyrinth) are irrelevant due to the small availability period. The advantages are highlighted by green background, the disadvantages by red one.

Some of the conflicts could be seen in that table: 4 out of the 5 most popular games have extreme long playtime for each puzzle (highlighted by red background). Because only a successfully finished gamelog can be used in cognitive evaluation, the long playtime for a puzzle means less data, the shorter puzzles are preferable. An example for that phenomenon is the Sudoku, the long total playtime Figure 3 and good daily playtime parameters are caused by the extra-long time needed to solve that type of puzzles. The situation is even worse, because in this table such short sessions are included as well, when the puzzle was given up after some seconds. The average value of 713 seconds (more than 10 minutes!)would be probably much higher if only the successful sessions were included.

GAME	NUMBER OF DAYS WHEN THE GAME WAS AVAILABLE	TOTAL PLAYTIME DURING THAT PERIOD (ALL PLAYERS) [SEC]	TOTAL PLAYTIME [HOUR]	DAILY PLAYTIME [HOUR]	NUMBER OF GAMELOGS	TIME/PLAYLOG [SEC]
LABYRINTH	22	764 605	212.4	9.7	2 784	274.6
FREECELL	398	11 639 364	3 233.2	8.1	32 777	355.1
SUDOKU	253	4 936 194	1 371.2	5.4	6 921	713.2
HIDDEN	118	1 621 630	450.5	3.8	34 081	47.6
LETTERS	260	3 277 594	910.4	3.5	9 180	357.0
PUZZLE	118	1 368 966	380.3	3.2	9 543	143.5
SWITCHPUZZLE	260	1 398 468	388.5	1.5	16 682	83.8
MEMORY	398	1 848 484	513.5	1.3	23 728	77.9
SEEKER	401	1 539 130	427.5	1.1	11 224	137.1
PLANAR	404	1 088 475	302.4	0.7	26 462	41.1
BLOCKS	365	880 855	244.7	0.7	18 750	47.0
HASHI	398	906 582	251.8	0.6	10 824	83.8
CONNECT	398	821 252	228.1	0.6	17 845	46.0
WGUESS	386	708 347	196.8	0.5	31 018	22.8
ROTATE	398	689 410	191.5	0.5	14 206	48.5

Figure 3 Total playtime of all users: most popular 15 games

The game Hidden (differences) seems be a good choice, but it has some drawbacks. It measures only one type of the cognitive ability (attention). Even more serious problem is that it is very hard to manage: new and new pictures should be given to sustain the entertainment capability.

In Figure 4 some other parameters and some other problems, contradictions of the 9 most popular games are shown. The time period of the evaluation was slightly different: from June 1, 2014 to March 31, 2015. Popularity is measured in this investigation by the number of game logs, which has primer importance in the measurement. However, it is different from the similar parameter of Figure 3, only the game logs containing successfully finished sessions are included (given up excluded). The games are sorted according to the "NUM. OF GAMELOGS (Given Up not included)" parameter. The advantages are highlighted again by green background, the disadvantages by red one.

In 7 of the games more than 10 thousand sessions were started (game logs were opened) during this 9 months period. But some games were too difficult for most of the players. For example FreeCell is one of the most popular ones, more than 16000 games were started, but only about 40% of the games were successfully finished, the other 60% were given up. Therefore less than 6300 could be evaluated – the measurement power of that game is seriously deteriorated by that fact.

Planarity was the most popular game in this period, but it has another drawback in measurement. The standard deviation of the playtime divided by the average playtime was very high compared to other games. This parameter gives a measure of randomness, therefore having this high noise; the measurement is significantly less reliable. The same phenomenon occurs in some other games as well: Connections, Hidden differences, Rotate are also problematic.

Word guess was popular, the given up ratio is nice, the noise is not extremely high during the measurements (but not really low), but the problem is that there are more than 200 possible settings (length of the words, keyboard setting, language, keyboard etc.), and about 65 of the settings were actually used. The different settings have to be evaluated as different games because there is no straightforward transformation between the results of different settings.

GAME	NUM. OF GAMELOGS OPENED (June 01, 2014-March 31, 2015)	GIVEN UP	GIVEN UP %	NUM. OF GAMELOGS (GIVEN UP NOT INCL.) (June 01, 2014-March 31, 2015)	NUM. OF SUBTYPES PLAYED	NUM. OF CORRECT LOGS (NO GIVEN UP) MOST FREQUENT SUBTYPE	NUM. OF PLAYERS (MOST FREQ. SETTING)	STDPlaytime / AVGPlayTime (BASED ON TOP 10 PLAYERS)
PLANAR	17 495	1 535	8.8	**15 960**	10	**10 736**	225	**0.84**
MEMORY	16 408	1 700	10.4	**14 708**	**22**	7 864	**257**	**0.30**
WGUESS	14 194	629	**4.4**	13 564	**~ 65**	8 167	162	0.57
BLOCKS	10 415	830	8.0	9 585	5	6 358	202	0.49
CONNECT	10 276	1 519	14.8	8 757	12	6 079	167	**0.65**
HIDDEN	14 563	6 002	**41.2**	8 561	1	8 561	227	**0.65**
PUZZLE	9 214	1 650	17.9	7 564	10	1 379	2	?
ROTATE	7 504	719	9.6	6 785	18	5 662	198	**0.69**
FREECELL	**16 059**	9 768	**60.8**	6 291	1	6 290	132	0.39

Figure 4 Parameters of the most popular 9 games, measured between June 01, 2014 and March 31, 2015

It could be stated that entertainment capability and measurement power are somehow contradictory requirements.

As it was shown the ideal game features:

- entertaining,
- small fluctuation in puzzle hardness,
- short solution time,
- easy to learn,
- easy to manage in the long run (automatic generation of very high number of puzzles),
- etc.

contradict each other. Therefore, the data processing and evaluation methods have to improve the change detection reliability.

1.4 Need for data fusion

Unfortunately, most people do not enjoy the same game for years. Therefore, in different time periods different games will be played by the same person. Not to destroy the level of motivation several games are offered (Figure 1); therefore, the performance measured using different games should be somehow compared to each other. In Figure 5 a player's performance measured using 3 different games is shown. One of the games was played in 4 different settings. Therefore, 6 different results should be fusioned somehow.

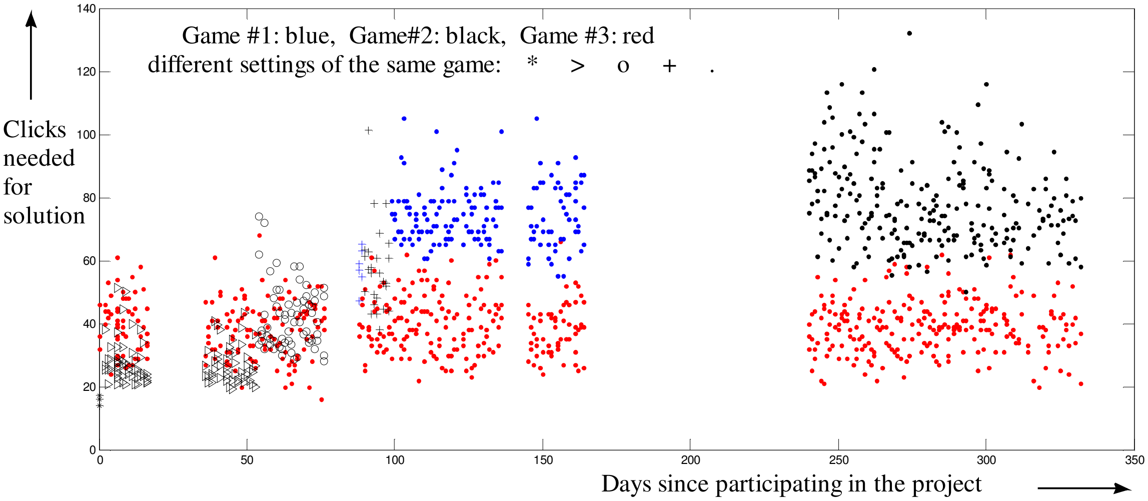


Figure 5 Typical example of a player's performance versus time. The time gaps are caused by travelling to relatives or by other reasons

This implies a sensor fusion and estimation problem, where the games are the sensors. It is similar to the modern pentathlon scoring problem, where performances in very different sports (fencing, show-jumping, running, swimming, shooting) have to be measured in one unified scoring scheme. In our case, the problem is even more complex because the same game could be played using different settings (e.g., different number of cards in the well-known memory game; therefore, each setting creates a new game from the measurement point of view). In the figure different games are marked by different colors; different settings of the same game are marked by different symbols. The proposed solution for solving this problem is detailed later (in). All these games should be compared to each other.

2 Data processing

Suggested data processing methods solve the problems analyzed in Section 1. Because the two-term noise is present, the effect of the impulse noise, the outliers, should be eliminated first.

2.1 Outlier detection method

There are several methods to detect (and eliminate) outliers.

One possible solution that the time between two consecutive elementary events during the solution (e.g. mouse clicks) is analyzed. Because the impulse noise is usually caused by an extreme interrupt, if the longest time between two such actions is too high in comparison to the average action time, then this game was probably seriously disturbed: it is taken as outlier and is rejected. This method was investigated during the project, but it was too complicated and slow.

During the project the method demonstrated in Figure 6 was implemented. Data are ordered and the three quartile values Q1, Q2, Q3 are determined. These three values divide the measured data population into four equal groups. Q1 is the value which is higher than the actual value of the 1/4th of the data. Q2 is the median value, which is higher than the actual value of half of the data, etc.

The inter quartile range IQ=Q3-Q1 contains half of the data. In this range the most typical values around the center are found. Outliers are defined above the upper limit UOF=Q3+3*IQ. In our case only outliers at one tail of the distribution should be detected, always the worse than typical case is interesting. In Figure 6 the high values are taken as outliers, for example the puzzle solving time could be such a parameter. If a disturbance occurs it always increases the time, therefore, the better than usual times are not taken as outliers. In case of other parameters it may happen that the low values are suspicious ones, in that case the LOF=Q1-3*IQ limit is used.

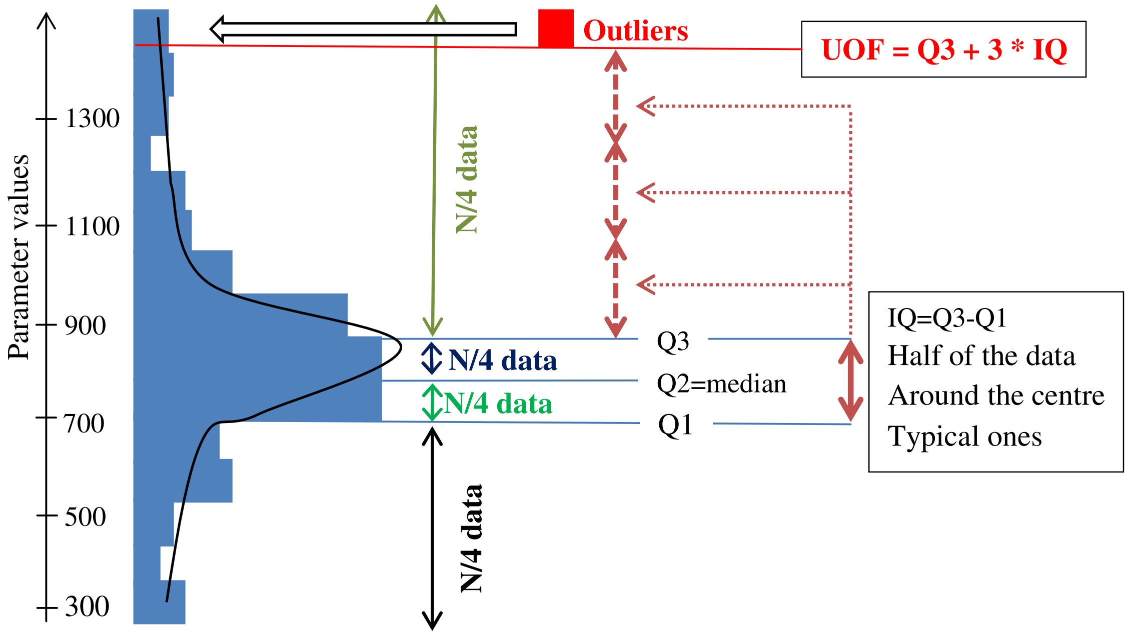


Figure 6 Outlier detection based on the distribution of the data measured

This method is better than the outlier detection algorithms using standard deviation of the data, because the standard deviation is more heavily influenced by the outliers than the quartile values.

2.2 *Solution to the problem caused by the noise term*

The other noise term, the small natural fluctuation must be coped with as well. For that reason, the change detection cannot be based on the performance measured in a single game; some sets of parameters should be compared. The goal is to detect the decline of performance, but in some periods improvements can occur as well. The assumption is that the decline is preceded by a period where no improvement is present; the situation is stable or deteriorating very slowly. Therefore, a reference set is selected, which is the group of consecutive games in which the person had stable performance (Figure 7).

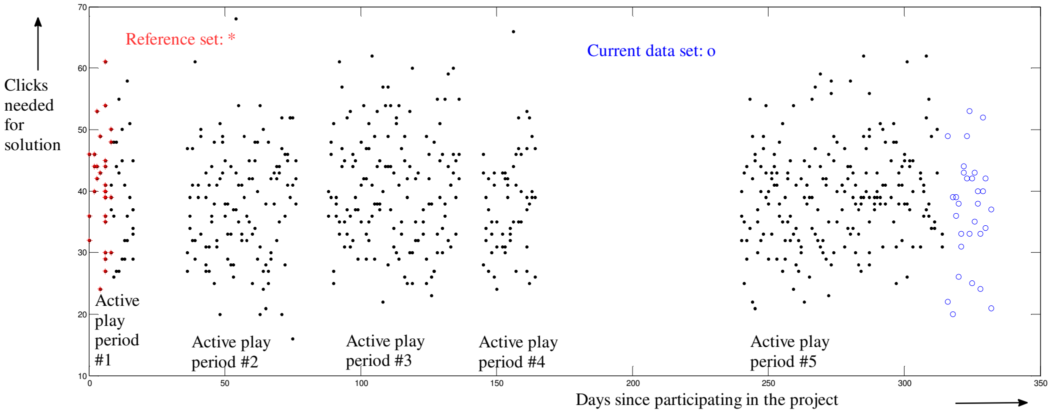


Figure 7 The current performance is always compared to the reference set

It is reasonably assumed that the short-term fluctuations due to tiredness, puzzle-hardness, etc., are zero-mean, stable independent random variables. The puzzle hardness is a zero-mean, stable random variable, because the same game is used with the same parameters, and the current puzzle is selected randomly. The short-term change of cognitive power is again a zero mean random variable, because it models the effects of the random changes of the environment, tiredness and health. The very slow long-term change of the cognitive state is modelled differently. Therefore, if a change is detected in one of the integral characteristics (mean, median, standard deviation) or generally in the distribution of the composite random variable (mental-state plus game-noise), it is caused by the slowly changing component modelling the mental state.

Let the performance observation based on the game played in time t_k be $\pi(t_k)$, k=1,2,…,K (this could be the score, the number of steps, etc.). Decrease in the values indicates decreasing performance in most of the cases. (There are some survival games, in which the longer playtime is better. In most of the cases the shorter ones.) Significant change in the time series cannot be stated while this seems to be a realization of an independent and identically distributed (i.i.d.) sample. Several statistical tests can be applied for testing the null hypothesis that the data is i.i.d. Such tests are the difference sign test, the turning point test and the rank test [1][2]. If the null hypothesis cannot be rejected, no significant change in the player's performance could be stated.

A less rigorous requirement is that we cannot justify a change, if the time series is weakly stationary; i.e., uncorrelated with constant expected value and variance. This null hypothesis can be tested with the Dickey-Fuller test [3]. If the time series seems to be non-stationary the change of the player's performance is detected.

Using the Mann-Whitney U or the Kolmogorov-Smirnov two-sample tests, the comparison of the distribution of the reference subset with the distribution of the currently examined subset of the time series could be performed. If we detect a difference between the distributions of the two sub-samples; and the current part of the series has smaller average (of ranks, of scores, etc.), then the player shows performance degradation.

These statistical hypothesis tests were used to check the distribution of the composite random variables. The tests were implemented in Matlab and SPSS. The following findings were obtained:

- The resulting performance parameter is not normally distributed according to the Lilliefors test. (Figure 8)
- The time gaps (several users produced 7…60 day gaps) did not change significantly the distribution of the random variable examined (see Table I).
- Several statistical tests were applied to compare the distribution of the reference period data to the current period data of users, who played some hundreds of games in the nearly one year period. (Two-sample Kolmogorov-Smirnov test, Mann-Whitney U test, Wilcoxon signed-rank test). The results confirm that both the stability and the change in the parameters are reliably estimated by the statistical tests. All these tests gave coherent results; later the performance of the different tests should be examined, and the best one should be selected.
- As an alternative to the two-sample statistical tests, a runs test on the sequence of observations was performed to prove the null hypothesis that the values came in random order, against the alternative that they did not. The runs test gave the same result: if there was no significant difference between the distributions of the reference and the

current subsets the runs test did not rejected the randomness hypothesis, if there was difference between them, the runs test rejected the hypothesis.

- In some cases, when starting a new game a learning phase occurs, in which the results are improving. The reference is meaningful only when the performance has stabilized. The stability could be defined the same way as the stationarity of the current performance. Evaluating these time series has proved that hypothesis testing detected the change of the cognitive performance as well.

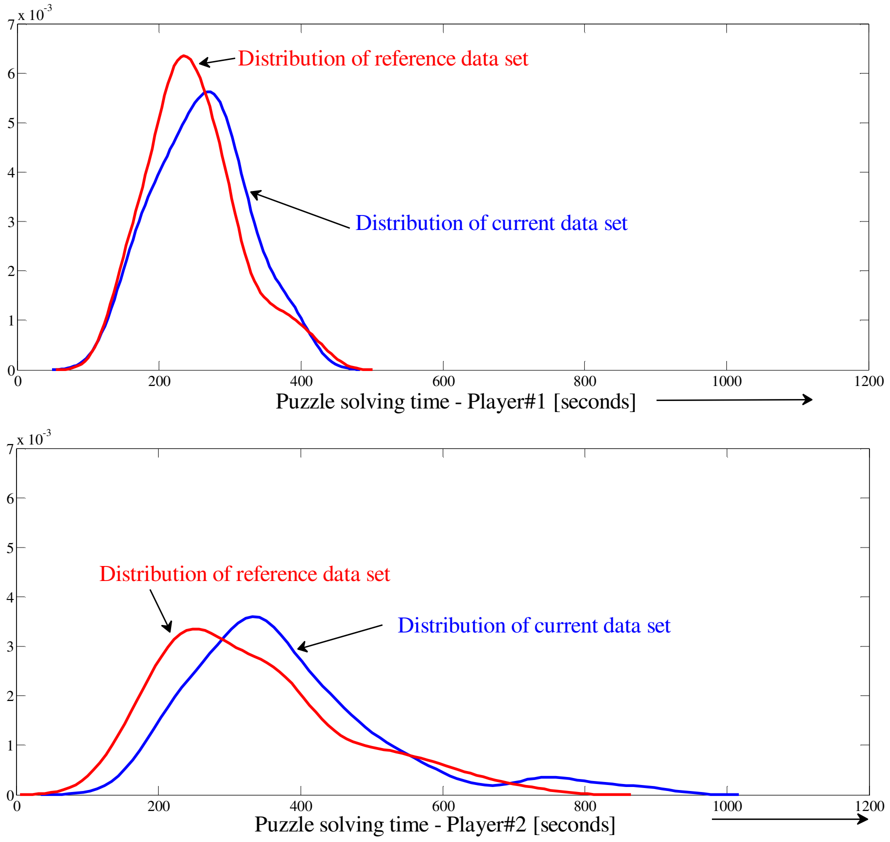


Figure 8 Two examples of puzzle solving time distributions

In Figure 9, a time series measured during the learning phase is shown. The hypothesis tests accepted the same distribution null hypothesis (the first 30 games' data compared to the current set) for all the sets up to the 187th game; and rejected the null hypothesis for all the sets from the 260th game.

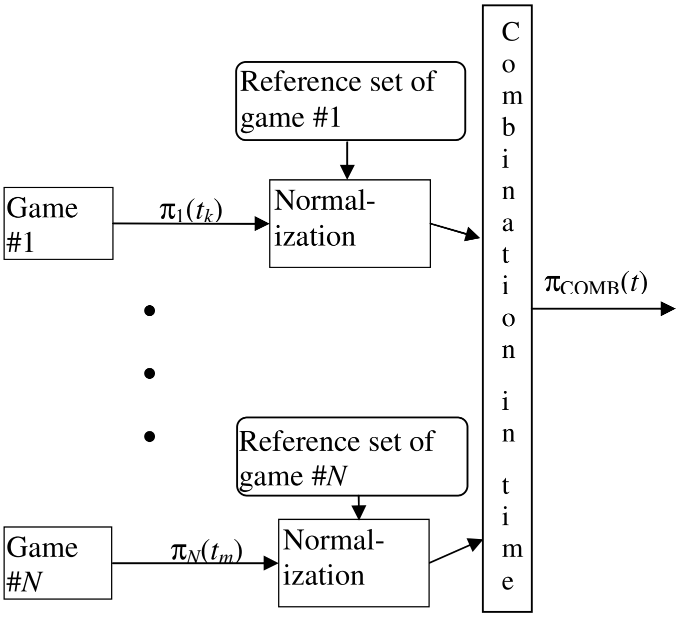


Figure 9 Nonstationary series of play times in the learning phase

2.3 Solving the sensor fusion problem

Computer games are proposed for detecting change in mental state as soon as possible. For motivational purposes several different games should be offered (and different settings of the same game could be used). Because of the voluntary nature there is no guarantee that the same person will play with the same game in the long-run. In our pilot only a few voluntary participants played continuously the same game for this nearly one year period. Most of them changed the game or at least changed the settings of a given game (to harder or to easier). Therefore, in different – overlapping and non-overlapping – time periods different sensors (games) are available to measure some parameters connected to mental state. Because the more data we have the more reliable the detection of the cognitive state; therefore, every effort is worth to keep all the data. In this section, two possible solutions of that sensor-fusion problem are proposed.

The basic idea of the first method is that proper linear normalization of the performance measures results in parameters, which are compatible with the normalized parameters of other games. The normalization is based on the reference set of the current game. Let the performance observation using game m in time t_k be $\pi_m(t_k)$, the average of the performance measures of this game's reference set be denoted by avg($\pi_{m\mathrm{REF}}$), the standard deviation of this reference set is std($\pi_{m\mathrm{REF}}$). The normalization:

$$\pi_{m\mathrm{n}}(t_k) \;=\; (\pi_m(t_k) - \mathrm{avg}(\pi_{m\mathrm{REF}}))/\mathrm{std}(\,\pi_{m\mathrm{REF}})\;,\; m=1,\ldots,N \qquad (1)$$

After normalizing all the parameters of the different games the combined time series is constructed by simply sorting the data in time.

$$\{\pi_{COMB\mathrm{n}}(t_1),\ldots,\pi_{COMB\mathrm{n}}(t_k)\}=\{\pi_{m1\mathrm{n}}(t_1),\ldots,\pi_{mk\mathrm{n}}(t_k)\,\}$$
$$t_1 < t_2 < \ldots < t_k \qquad (2)$$

The block diagram of the suggested idea is shown in Figure 10.

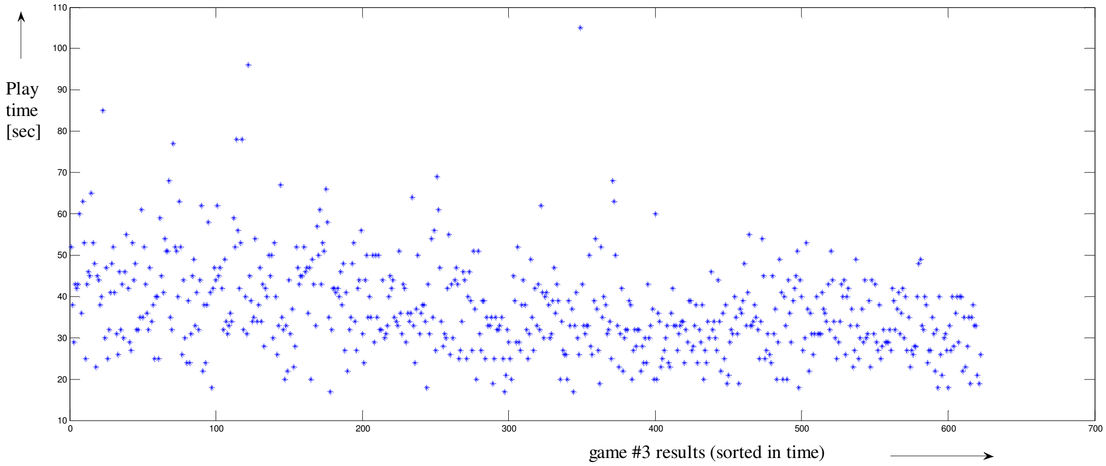


Figure 10 Normalized performance parameters of the different games are combined using linear normalization to form one composite time series

The resulting combined time series derived from the data of Figure 2 is shown in Figure 11.

After normalization the two-sample Kolmogorov-Smirnoff hypothesis test was used two reject or accept the "two distributions are identical" hypothesis. Using the time series of combined data gives very similar results as using the data of one game only. In Table I the null hypothesis of having the same distribution of the data subsets compared are shown in two ways. In both evaluations the reference set comes from the first 30 observations of the active play period 1, the comparison is made to the first 30 observations of the 2nd, 3rd, 4th, 5th active play periods, respectively. The difference is that in the first experiment only the Game#3 data are used, and in the second experiment the combined data are used.

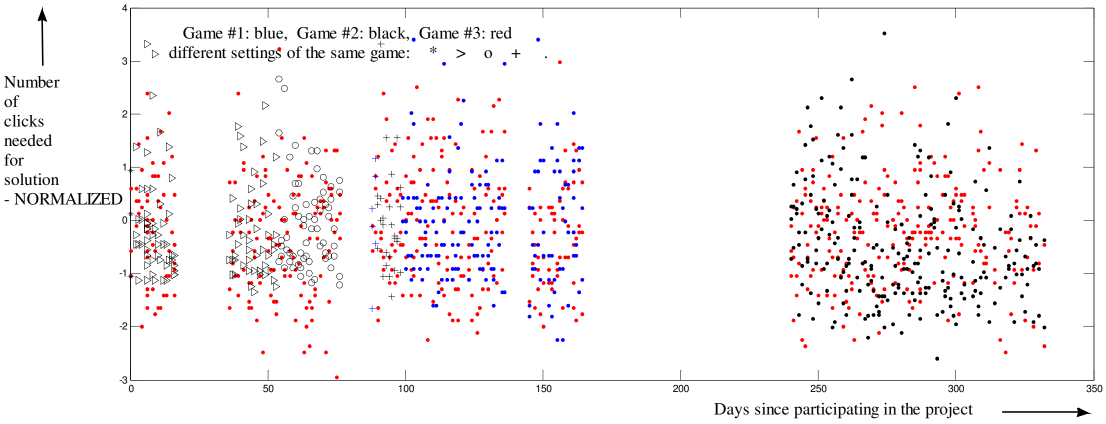


Figure 11 Normalized and combined data

In Table I, the acceptance or rejection (on the p=0.05 level) of the null hypothesis are shown.

TABLE I. RESULTS OF TWO-SAMPLE KOLMOGOROV-SMIRNOV TESTS: REFERENCE SET SHOWN IN FIGURE 5 (FIRST 30 OBSERVATIONS OF ACTIVE PLAY PERIOD 1) COMPARED TO THE FIRST 30 DATA OF EACH ACTIVE PLAY PERIOD

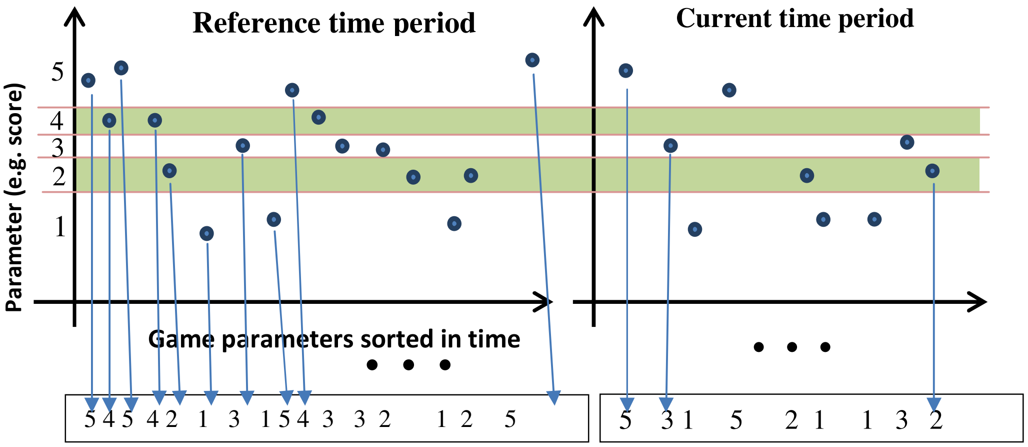
	Game #3 data only		***Combined data***	
Reference: active play period 1 compared with	***Null hypothesis accepted:0, rejected: 1***	***Probability value:*** ***p***	***Null hypothesis accepted:0, rejected: 1***	***Probability value:*** ***p***
Active play period #2	0	0.43	0	0.11
Active play period #3	0	0.76	0	0.20
Active play period #4	1	0.03	1	0.01
Active play period #5	0	0.54	0	0.06

Although in the first experiment only Game#3 data were used and in the second one combined data were used, they resulted in the same acceptance/rejection scheme although the pure one-game only data gave higher probability values.

The basic idea of the second method uses quantization based on the distribution of the reference period data. Based on the reference data set, 4 limits are defined, which divide the set to 5 equal groups. These groups are represented by the integers 1, 2, 3, 4, 5 respectively. The best performance corresponds to 5, the worst one to 1. These limits are frozen and they are used to classify the data of the current set as well. If the distribution has not changed, the integers characterizing the current set data will be nearly equally distributed as well. If the performance improved, then more 4 and 5 will be in the current set than 1 and 2. If the performance deteriorated, then more 1 and 2 will be in the current set than 4 and 5.

Using that method every parameter set is transformed into a set of integers. This nonlinear transform is a quantization with very few steps.

The method is summarized in Figure 12.



Limits: in all the 5 groups (nearly) the same number of data

Same limits are used, different number of data could be in different groups

Figure 12 Standardization using quantization

This method results in a standardized series, in which the labels 1,2,…,5 always have the same meaning. Therefore, results of different games and different settings could be directly combined or compared.

The standard chi-square test could be used to check whether the “two distributions are identical” hypothesis could be rejected or not.

During the project both above mentioned methods were tested. The first method proved to be too sensitive, too many false alarms were detected. Therefore, the second method (standardization with quantization) was implemented.

3 Data processing and evaluation algorithm

3.1 The implemented algorithm

The implemented algorithm contains the methods detailed in Section 2. The algorithm is summarized in Figure 13.

1. Each day the new game logs are tested. All the players in the player database are tested for all games, whether in that day this player successfully finished a new game session (puzzle) or not.
2. If there was a successful (not given up) new game, then the current set of 50 game log parameters and the reference set of 50 game log parameters are created. The current set is simply formed from the parameters of the last 50 game logs in time. (The creation of the reference set is detailed later.)
3. The outliers are detected and cleaned from the datasets.
4. The data are standardized (normalized) using the quantization method detailed in Section 2.
5. If more than one game were played, the results are combined. (The possible combinations to be checked could be given in a setup file when the system is implemented.)
6. Statistical hypothesis test (chi-square test) is performed: could we reject the hypothesis that the two distributions are the same? The theoretical considerations and the experience gathered show that both sets should contain about 50 valid data.

   If the hypothesis could be rejected the average of the current set is analysed. (The reference set has always an average close to 3, because the normalization creates 5 equal groups of 1, 2,… ,5.) If the current set's average is higher than 3 (and the identical hypothesis is rejected), then the performance improved. If the average is lower than 3, the performance deteriorated.

7. The result of the hypothesis test is written to the database.

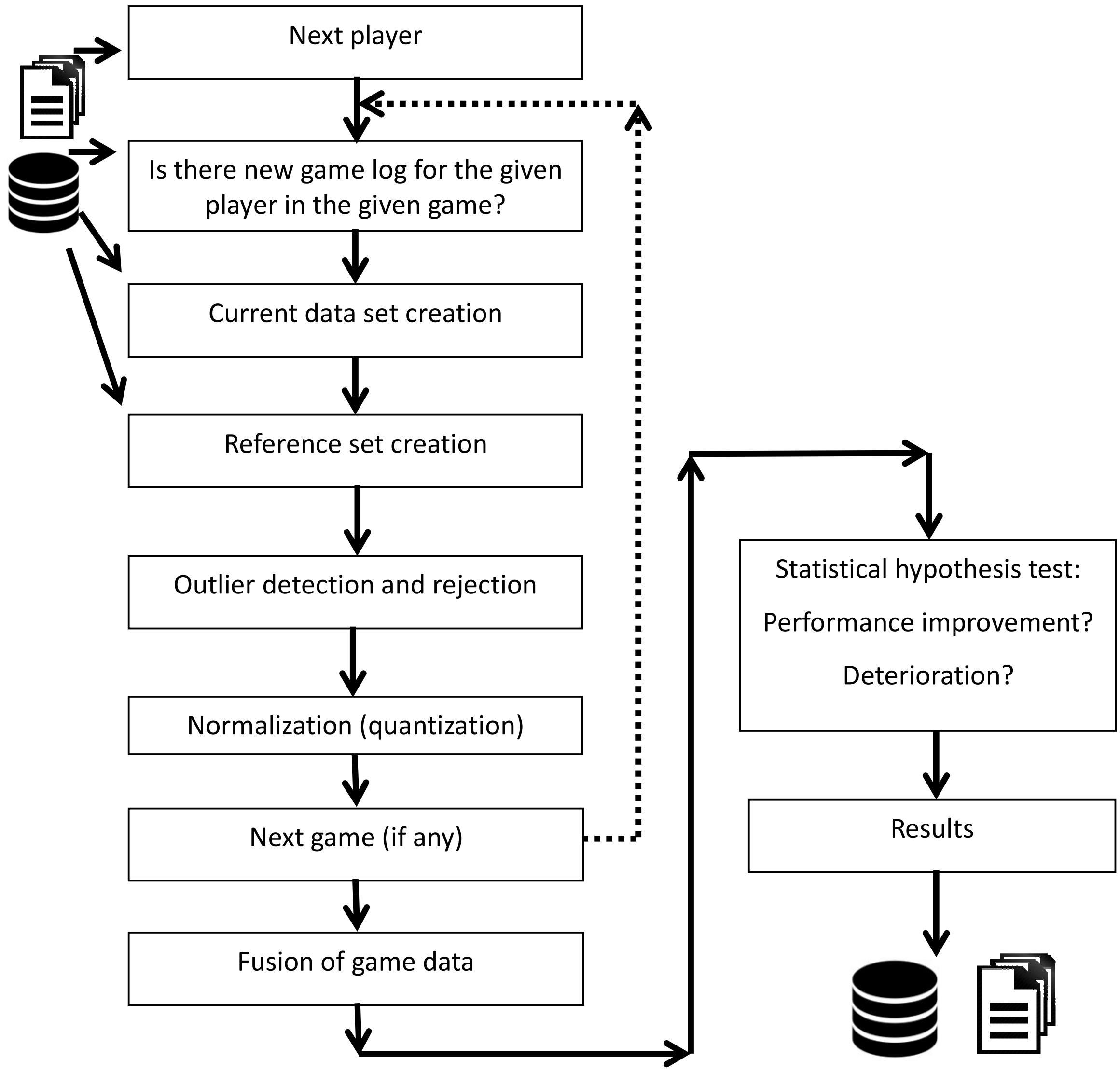

Next player
Is there new game log for the given player in the given game?
Current data set creation
Reference set creation
Outlier detection and rejection
Normalization (quantization)
Next game (if any)
Fusion of game data
Statistical hypothesis test:
Performance improvement?
Deterioration?
Results

Figure 13 Data processing and evaluation algorithm

The creation of the current data set is straightforward. Starting with the latest session the algorithm collects the previous game logs, until the 50 data are reached. (Actually it could be a bit more game logs, if there are outliers.) The creation of the reference set is more problematic. The first 50 game logs could be used after the player registration in many cases. (Of course the outliers have to be cleared.) The situation is not so simple if the player start to learn a new game, because his/her performance will improve at first. If we used the first 50 game logs, than the reference would show worse performance than the real one (see Figure 14).

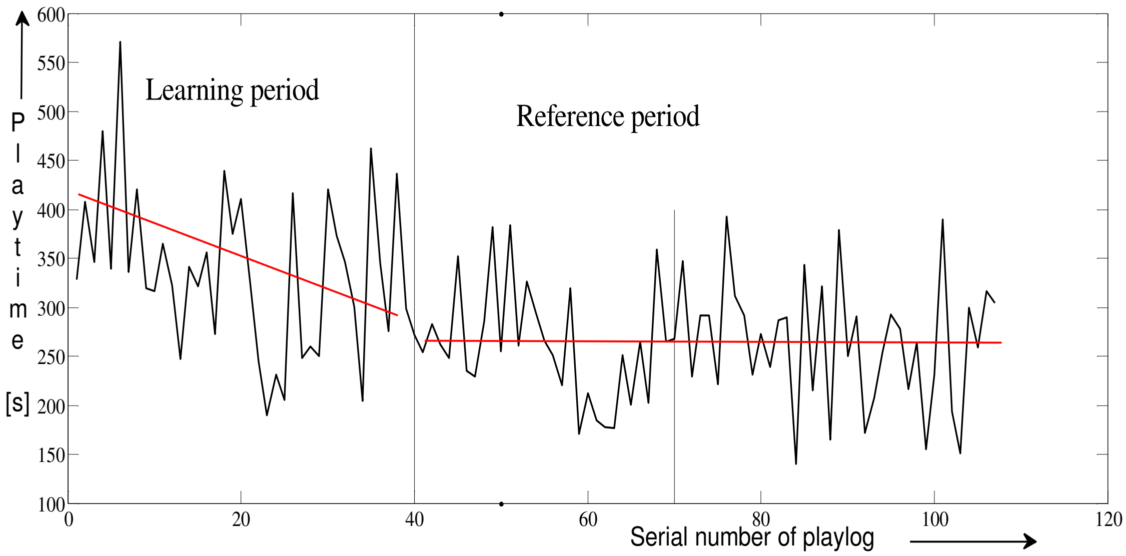


Figure 14 Typical learning effect

Therefore, the reference must be the first 50 game logs after the performance stabilized.

3.2 Demonstrative actual results

The algorithm detailed in section 3.1 was tested on game logs of several thousand sessions in 2014 and 2015. Typical results are shown in Table II. These were the results of the test of July 14, 2015. Some results are shown in Table II, the whole set of results is given in Appendix I. There were more tests started than these 25 ones, but either the reference set or the current set was not large enough, or the time gap between them was not long enough.

In the first column the player identification numbers, in the second row the game identification numbers are shown. In this test only single games were tested, but game combinations to be tested could be defined as well.

The start and end dates of the reference period depend on the end of the learning period and on the playing frequency. Player 5217 played so much with game 180 (Memory or Find the pairs) that he/she produced the 50 sessions needed for reference set in 24 hours.

In the last 3 columns the results of the statistical tests are given. 95% level was set, therefore the null hypothesis is rejected if p < 0.05. If the hypothesis rejected, then the averages of the reference set and the current set are analysed. If the average increased then the current performance is better than it was in the reference period.

TABLE II. SOME OF THE DEVELOPED ALGORITHM'S RESULTS ON JULY 14, 2015

PLAYER ID	GAME CODE	REF FIRST DATE	REF LAST DATE	TEST FIRST DATE	TEST LAST DATE	REF. DISTRIBUTION	TEST DISTRIBUTION	P-VALUE	IS REJECTED	IS BETTER
5217	(180)	2015-05-23	2015-05-24	2015-07-11	2015-07-13	[9, 10, 10, 10, 11]	[4, 20, 6, 2, 8]	0.03	true	false
3426	(150)	2015-05-21	2015-06-03	2015-07-	2015-07-13	[9, 10, 10, 10, 11]	[0, 5, 5, 13, 17]	0.01	true	true
235	(150)	2015-05-28	2015-06-06	2015-07-09	2015-07-13	[9, 10, 10, 10, 11]	[4, 5, 2, 2, 27]	0.00	true	true
269	(290)	2014-12-25	2015-02-04	2015-06-10	2015-07-13	[9, 10, 10, 10, 11]	[8, 6, 4, 13, 9]	0.53	false	-
269	**(310)**	**2015-03-26**	**2015-04-22**	**2015-06-20**	**2015-07-13 20**	**[9, 10, 10, 10, 11]**	**[1, 0, 4, 6, 29]**	**0.00**	**true**	**true**

The performance of player 269 in game 310 (Blocks) is demonstrated in Figure 15. The start and end dates of reference set and current set was shown in Table II as well. Because significant improvement occurred in the first 130 games, the algorithm set the reference period from the 131th game to the 18th game. The quantization levels were set such a way, that the 1 label occurred 9 times, the 2 occurred 10 times, the 3 occurred10 times, the 4 occurred 10 times, and the 5 occurred 11 times in the reference series. The end of current set was defined (the last game in July 13, 2015); and the last 40 games were used. The same quantization levels gave for the current set 1 in 1 case, 2 was not given to any session, 3 in 4 cases, 4 in 6 cases and 5 in 29 cases. Therefore the statistical test rejected the null hypothesis, and improvement of the performance was detected. It shows that after the performance stabilized a slow improvement occurred.

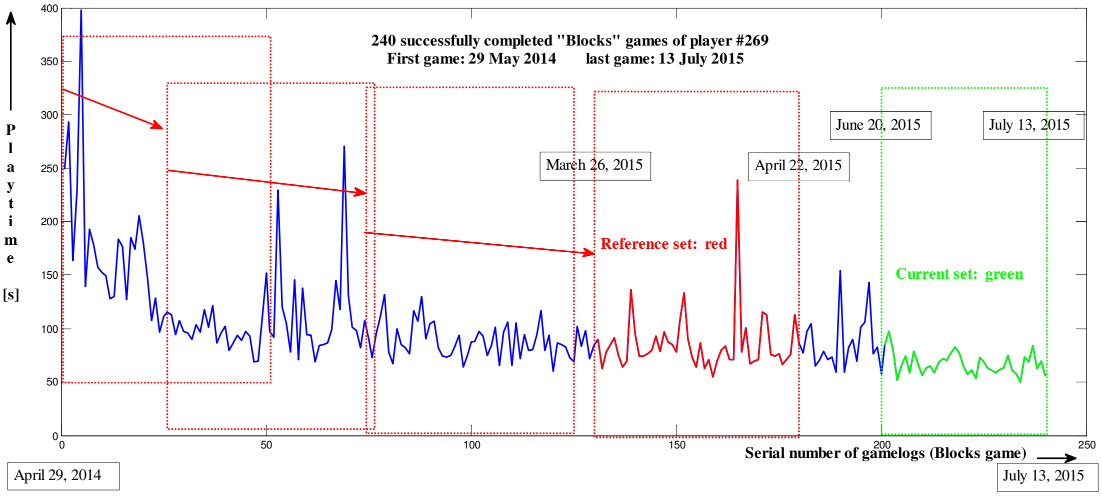


Figure 15 Reference set and current set in the analysis of the performance of player 269 shown in Blocks game. The reference set was iteratively moved until the performance stabilized.

4 The implemented StatEval program

4.1 Overview

The eu.m3w.stateval is a statistical evaluation software tool, developed in the M3W project (M3W - Maintaining and Measuring Mental Wellness). The purpose of this tool is to examine the game logs of the players and detect the long term changes in a player's performance. The examination is done separately for all players and groups of games, by selecting two segments of the time series of the game logs, called the Reference and the Test. These two sets of game logs are normalized, and compared with a statistical test.

4.2 Terminology

Tool The eu.m3w.stateval statistical evaluation tool.

Game combination A group of games that are used together in a reference or test set.

Reference The set of game logs for a specific player and game combination that represents the baseline performance of the given player in the given games. It is taken from the past, usually from the earliest game logs of the player.

Test The set of game logs for a specific player and game combination that represents the current performance of the given player in the given games. Usually these are the newest game logs of the player.

Reference date The first game log in the Reference must not be older than this date.

Till date The newest game log in the Test must not be newer than this date.

From date The start of the time interval of new game logs. All game logs between the From date and Till date will be added to the Test.

Database The M3W Data Service. https://m3w.mit.bme.hu/ds/

Source The source where the input game logs of the Tool are taken from. It can be either the Database, or a directory with log files.

Output The destination where the output of the statistical analysis is put. It can be either the Database or a CSV file.

Configuration file The file that contains all configurations. Default path is 'config.json'.

4.3 Outline of operation

1) Determine the Source from the Configuration file. Source can be the Database, or can be a directory with game log files.
2) Determine Till date, From date and Reference date
   a) If the Source is log files, and the *isLogsFromFilesUseDates* setting is false, the Till date is the maximal UNIX date, the From date is the same as Till date. Reference date is the zero UNIX date.

b) If some, or all of the dates are specified as command line arguments, they are used, instead of the default values.

c) If nothing is specified, Till date is the current date, From date is 1 day before Till date, and Reference date is the zero UNIX date.

3) Determine which players and for which games have new game logs, by querying the Source. All players in all game combinations where they have at least one new game log will be tested.

a) If the Source is log files, and the *isLogsFromFilesUseDates* setting is false, all log files are considered.

b) Otherwise the Source is queried for game logs between the From and Till dates.

4) For each player and each game combination, where there are new game logs, the Test is queried from the Source.

a) The Source is queried for some game logs for the given player and given game combinations

b) Outlier filtering is performed on the received game logs.

c) 4)a)-4)c) is repeated until there are at least *sampleLength* number of non-outlier game logs and the last game log is older than From date.

d) If there are not enough game logs for Test, the given analysis is discarded

5) After the Test is determined for a player and a game combination, the Reference is determined.

a) The Source is queried for some game logs for the given player and given game combinations between the Reference date and *refTestMinGap* time before the first game log date in Test.

b) Outlier filtering is performed on the received game logs.

c) 5)a)-5)c) is repeated until there are at least *referenceSampleLength* number of non-outlier game logs.

d) The Reference is normalized

e) If there are less than *referenceMinSampleLength* normalized game logs in Reference, the given analysis is discarded

6) The Test is normalized with the same settings as the Reference
7) The statistical analysis is performed on the Reference and the Test
    a) If the Test is significantly better than the Reference, the Reference date for this analysis is set for the date of the last game log in the Reference, and 5)-7) is repeated (i.e.: the Reference is shifted to the next *referenceSampleLength* game logs), unless the Reference would get closer to the Test than *refTestMinGap*.
    b) Otherwise the result of the analysis is written to the Output.
8) If batch mode is specified, the From, Till and Reference dates are adjusted, and 3)-7) are repeated until *batchCount* iterations are performed.

4.3.1 Outline of the outlier filtering algorithm

1) The input is a set of game logs from possibly different games and games with different settings in a game combination.
2) The input game logs are grouped into homogeneous sets, in which each game log is from the same game and has the same game settings.
3) Each of these homogeneous groups is filtered for outliers on their own. For this, a couple of parameters are calculated first.
    a) If the homogeneous group has less than *outlierFilterMinLength* game logs, all game logs in that group is considered outlier, and the group is discarded for further outlier filtering.
    b) If a game log is *isGivenUp* , that game log is considered outlier, and is not considered further
    c) For each parameter, the *quartiles Q1*, *Q2* and *Q3* are calculated. The *inter-quartile range* is calculated, *IQR=Q3-Q1*. The *lower* and *upper outer fences* are calculated: *LOF=Q1-3*IQR*, *UOF=Q3+3*IQR*.
4) Each game log in the homogeneous group is checked if being an outlier
    a) Each allowed parameter of the game log is filtered independently for outliers (score, playtime, additional parameters. Filters that allow/deny these: *gameScoreFilter*, *gamePlayTimeFilter*, *gameAdditionalParamsFilter*)

b) It is determined for the given parameter that the better performance is described with higher or smaller values

c) If a given parameter is outside the *outer fence* in the direction of worse performance, the game log is considered outlier.

4.3.2 Outline of the normalization algorithm

1) The input is a set of game logs from possibly different games and games with different settings in a game combination. The normalizer is initialized with another set of game logs, usually the Reference.

2) The input game logs are grouped into homogeneous sets, in which each game log is from the same game and has the same game settings.

3) Each of these homogeneous groups is normalized on their own. For this, a couple of parameters are calculated first.

4) If *normalizerClass* is *NormalizerUniform*

   a) Each allowed parameter of the game log is normalized independently (score, playtime, additional parameters. Filters that allow/deny these: *gameScoreFilter*, *gamePlayTimeFilter*, *gameAdditionalParamsFilter*)

   b) The initialization set is used to calculate the limits for all parameters, so they are uniformly distributed in five bins, 1-5.

   c) The normalized value of a game log (the index of the bin, 1-5) is the average of the normalized parameter values.

5) If *normalizerClass* is *NormalizerOld*

   a) Each allowed parameter of the game log is normalized independently (score, playtime, additional parameters. Filters that allow/deny these: *gameScoreFilter*, *gamePlayTimeFilter*, *gameAdditionalParamsFilter*)

   b) The initialization set is used to calculate three points in the parameter space, the *good*, the *average* and the *bad*, which are the maximal, average and mean values of the parameters of the initialization set of game logs.

c) The normalized value of a game log depends on its distance from the three points. If it is closest to the *good*, it will be 5, if it is closest to the *bad*, it will be 1. If it is closest to the *average*, and closer to the *bad* than the *good*, it will be 3, otherwise it will be 4.

4.3.3 The statistical test

The statistical test is a Chi-squared test. It is used to test whether the two discrete sample sets come from the same distribution, where the two sets are the normalized Reference and normalized Test game log sets. Example: Reference: [9 10 10 10 11], Test: [8 6 12 7 7]. After the statistical test is done, the expected value (average) of both the Test and the Reference are calculated. If the statistical test resulted in rejecting the null hypothesis, thus showing that the two sets are significantly different, the expected values are compared. If the Test is larger, it means that the performance of the Player on that specific Game combination has been improved. On the other hand if the Reference is larger, it means that the Player performance has been dropped.

4.3.4 Output format

The Tool can write the results into two kinds of output, into a CSV file or into the Database.

4.3.4.1 CSV file format

Each row in the CSV file is a result of an analysis. Each row has the following columns.

- PLAYER – the player ID of the player of the game logs in this analysis
- GAME COMBINATION – the game IDs in this analysis
- REF FIRST DATE – the date of the first game log in the reference
- REF LAST DATE – the date of the last game log in the reference
- TEST FIRST DATE – the date of the first game log in the test
- TEST LAST DATE – the date of the last game log in the test
- REFERENCE DISTRIBUTION – five integer numbers, the count of game logs in each bin in the Reference

- TEST DISTRIBUTION – five integer numbers, the count of game logs in each bin in the Test
- P-VALUE – the p-value of the Chi-squared test of the Reference and Test
- IS REJECTED – it is true, if the null hypothesis is rejected
- IS BETTER – it is true, if the Test has larger expected value than the Reference

4.3.4.2 Database fields

Each record in the *analysisresults* table is the result of an analysis. The table has the following columns.

- *playerid* (INT) – the player ID of the player of the game logs in this analysis
- *gamecombination* (VARCHAR) – the game IDs in this analysis
- *options* (VARCHAR) – the settings this analysis was performed with. It has the following settings. Look for settings meanings in the Configuration chapter.
    - *sampleLength* (SL)
    - *referenceSampleLength* (RSL)
    - *referenceMinSampleLength* (RMSL)
    - *normalizeMinLength* (NML)
    - *outlierFilterMinLength* (OFML)
    - *minBinValue* (MBV)
    - *significanceLevel* (SgL)
    - *scoreFilter* (SF) (Y-yes, N-no, X-some games yes, some games no)
    - *playtimeFilter* (PTF) (Y-yes, N-no, X-some games yes, some games no)
    - *additionalParametersFilter* (APF) (Y-yes, N-no, X-some games yes, some games no)
- *referencefirstdate* (DATE) – the date of the first game log in the reference
- *referencelastdate* (DATE) – the date of the last game log in the reference
- *testfirstdate* (DATE) – the date of the first game log in the test
- *testlastdate* (DATE) – the date of the last game log in the test

- *referencedistribution* (VARCHAR) – five integer numbers, the count of game logs in each bin in the Reference
- *testdistribution* (VARCHAR) – five integer numbers, the count of game logs in each bin in the Test
- *pvalue* (FLOAT) – the p-value of the Chi-squared test of the Reference and Test
- *isrejected* (BOOLEAN) – it is true, if the null hypothesis is rejected
- *isbetter* (BOOLEAN) – it is true, if the Test has larger expected value than the Reference

4.4 *Configuration*

4.4.1 Command line arguments

--help	-?	Displays a brief summary of all the command line arguments and configuration file options.
--configfile=	-c=	Name of the configuration file. Default: config.json
--date=	-d=	Till date. This is also the end of the time interval that is checked for new game logs. Default: now. Format: YYYY-MM-DD_hh:mm:ss
--dateFrom=	-dF=	From date. This is also the start of the time interval that is checked for new game logs. Default: Till date - 1 day
--dateRef=	-dR=	Reference date. The start of the time interval that is checked for reference game logs. Default: 1970-01-01 00:00:00 (UNIX zero time)
--debug		If specified, all intermediate analysis data is written into .csv files for debug purposes. Default: off

--verboseLevel=	-v=	Sets the verbose level. 0 for no messages, 1 for main messages only, 2 for all messages. Default: 0
--batchCount=	-bc=	The number of iterations to perform the analysis and increment the specified dates with the specified timespans. Default: 1
--batchTillIncrement=	-bdi=	The timespan to increment the Till date. Format: 1[d, h, m, s] for days, hours, minutes and seconds, respectively. Without postfix, the number is considered as milliseconds. Example: 1d Default: 0s
--batchFromIncrement=	-bfi=	The timespan to increment the From date. Default: 0s
--batchRefIncrement=	-bri=	The timespan to increment the Reference date. Default: 0s

4.4.2 Configuration file

The configuration file is a JSON object, with the following fields.

4.4.3 Input/Output configuration

logsFromFiles

Determines whether the input source is a directory of game log files or not (i.e.: it is the database).

JSON type: Boolean

Optional

Default: false

logsDir	The path to the log files directory. JSON type: String Mandatory, if *logsFromFiles* is true Example: '../logs'
logsFromFilesUseDates	Determines if the Reference from, From and Till dates should be used for the Test and Reference sets selection, or all game logs should be considered in the specified directory for this purpose. JSON type: Boolean Optional, only applies when *logsFromFiles* is true Default: false
databaseCacheDisabled	Determines whether the database caching should be skipped. Not recommended, especially for batch use, as single database queries are very slow compared to caching the whole database. JSON type: Boolean Optional, only applies when *logsFromFiles* is false Default: false
outputToFile	Determines whether the output should be written to a CSV file instead of written to the database. JSON type: Boolean Optional Default: false, if *logsFromFiles* is true, it is automatically set to true, as it is forbidden to write the output of arbitrary input to the database.
outputFilename	The path to the output CSV file. JSON type: String Optional, only applies when *outputToFile* is true Default: 'output.csv'

CSVSeparatorValue

The separator character between the fields in the CSV file.
JSON type: String
Optional, only applies when *outputToFile* is true
Default: ‘,’ (comma)

4.4.4 Statistical analysis configuration

sampleLength

The required amount of game logs in the Test, after outlier detection has been performed. Actually more game logs can be in the Test when there are more game logs between the From and Till dates that *sampleLength*. If only less game logs are available as Test, the given analysis is aborted. Less game logs can also be in the Test, when in some homogeneous game sets the *normalizeMinLength* condition is not met.
JSON type: Integer
Mandatory
Example: 30

referenceSampleLength

The required and maximal number of game logs in the Reference, after outlier detection has been performed. Less game logs can also be in the Reference, when in some homogeneous game sets the *normalizeMinLength* condition is not met, or when only less game logs are available as Reference.
JSON type: Integer
Mandatory
Example: 50

referenceMinSampleLength	The required minimal number of game logs in the Reference, after outlier detection and normalization has been performed. If there are less game logs in the Reference, the given analysis is aborted. JSON type: Integer Mandatory Example: 30
normalizeMinLength	The required minimal number of game logs in a homogeneous game set required performing the normalization of the given game set. If there are less game logs in the game set, that set is not normalized. JSON type: Integer Mandatory Example: 5
outlierFilterMinLength	The required minimal number of game logs in a homogeneous game set required performing the outlier filtering of the given game set. If there are less game logs in the game set, that set is not outlier filtered. JSON type: Integer Mandatory Example: 5
minBinValue	The required minimal number of game logs in a normalized bin in the Reference. If there are fewer game logs in the bin, it gets combined with a neighboring bin. JSON type: Integer Optional Default: 5

significanceLevel

The significance level (or 1 – p-value) above which the null hypothesis is rejected, that the Test and Reference samples are drawn from the same distribution.

JSON type: Float

Mandatory

Example: 0.95

normalizerClass

The algorithm to use for game set normalization.

JSON type: String, ‘NormalizerUniform’ or ‘NormalizerOld’

Optional

Default: ‘NormalizerUniform’

refTestMinGap

The minimal time interval between the last Reference game log and the first Test game log.

JSON type: String, time span, 1[d, h, m, s] for days, hours, minutes and seconds, respectively

Optional

Default: ‘0s’

4.4.5 Analysis target configuration

gameCombinations

The list of game combinations that will be used for the analysis.

JSON type: map, where key is the game combination as string, and the value is the custom query URL or empty string for default query URL.

Mandatory

Example: { '(190, 130, 250)' : '/ds/customQuery', '(100)' : '' }

playerFilter	Only the filtered player ID-s will be analysed. JSON type: Map, which is a filter, see below Optional Default: { generalIsIncluded : true, exceptions : [] } Example: { generalIsIncluded : true, exceptions : [0, 1] }
gameScoreFilter	The score of the game logs of filtered games will be considered. JSON type: Map, which is a filter, see below Optional Default: { generalIsIncluded : false, exceptions : [] } Example: { generalIsIncluded : true, exceptions : [100] }
gamePlayTimeFilter	The play time of the game logs of filtered games will be considered. JSON type: Map, which is a filter, see below Optional Default: { generalIsIncluded : true, exceptions : [] } Example: { generalIsIncluded : false, exceptions : [190] }
gameAdditionalParamsFilter	The additional parameters (if any) of the game logs of filtered games will be considered. JSON type: Map, which is a filter, see below Optional Default: { generalIsIncluded : false, exceptions : [] } Example: { generalIsIncluded : true, exceptions : [130, 250] }

4.4.6 Filter

Filters are used to filter different objects from collections. They have a general rule that implicitly applies to every object, and exception rules that explicitly applies to the enumerated objects.

generalIsIncluded	Determines whether every object that is not explicitly listed as exceptions, will be filtered, or not. JSON type: Boolean Mandatory
exceptions	The explicit list of exception objects, that will be inversely filtered as every other objects in general. JSON type: List Mandatory

References

[1] P. Brockwell and R. Davis, Introduction to Time Series and Forecasting, Springer, New York, 1996.

[2] M. G. Kendall and A. Stuart, "The Advanced Theory of Statistics", Vol. 3, Griffin, London, 1976.

[3] D. A. Dickey, W. A. Fuller, "Distribution of the Estimators for Autoregressive Time Series with a Unit Root". J. of the American Statistical Association 74 (366), Jun. 1979, pp. 427–431.

[4] P. Breuer, P. Hanák, L. Ketskeméty, B. Pataki, G. Csukly, "Home Monitoring of Mental State With Computer Games: Solution Suggestion to the Mental Modern Pentathlon Scoring Problem", Proc. of The Eighth International Conference on Advances in Computer-Human Interactions. Lisbon, Portugal, Febr. 22-27, 2015 2015. pp. 1-7

[5] B. Pataki , P. Hanák, G. Csukly, "Computer Games for Older Adults beyond Entertainment and Training: Possible Tools for Early Warnings: Concept and Proof of Concept", Proc of the International Conference ICT4AgeingWell 2015. Lisbon, Portugal, May 20-22, 2015 pp. 285-294.

Appendix

The results of tests on July 14, 2015.

PLAYER ID	GAME CODE	REF FIRST DATE	REF LAST DATE	TEST FIRST DATE	TEST LAST DATE	REF. DISTRIBUTION	TEST DISTRIBUTION	P-VALUE	IS REJECTED	IS BETTER
2464	(150)	2015-05-19	2015-06-03	2015-07-04	2015-07-13	[9, 10, 10, 10, 11]	[0, 1, 4, 5, 30]	0.00	true	true
5217	(180)	2015-05-23	2015-05-24	2015-07-11	2015-07-13	[9, 10, 10, 10, 11]	[4, 20, 6, 2, 8]	0.03	true	false
3426	(150)	2015-05-21	2015-06-03	2015-07-10	2015-07-13	[9, 10, 10, 10, 11]	[0, 5, 5, 13, 17]	0.01	true	true
235	(150)	2015-05-28	2015-06-06	2015-07-09	2015-07-13	[9, 10, 10, 10, 11]	[4, 5, 2, 2, 27]	0.00	true	true
269	(290)	2014-12-25	2015-02-04	2015-06-10	2015-07-13	[9, 10, 10, 10, 11]	[8, 6, 4, 13, 9]	0.53	false	true
269	(150)	2014-05-29	2014-08-19	2015-06-12	2015-07-13	[9, 10, 10, 10, 11]	[4, 7, 8, 5, 16]	0.38	false	true
269	(310)	2015-03-26	2015-04-22	2015-06-20	2015-07-13	[9, 10, 10, 10, 11]	[1, 0, 4, 6, 29]	0.00	true	true
269	(130)	2014-10-29	2015-04-29	2015-06-04	2015-07-13	[9, 10, 10, 10, 11]	[1, 6, 10, 14, 9]	0.12	false	true
5904	(190)	2015-05-23	2015-06-01	2015-07-07	2015-07-13	[9, 10, 10, 10, 11]	[0, 0, 2, 2, 4]	0.26	false	true
16	(150)	2014-05-29	2014-11-05	2015-04-12	2015-07-13	[9, 10, 10, 10, 11]	[3, 4, 5, 10, 18]	0.09	false	true
2257	(270)	2015-03-15	2015-03-20	2015-06-25	2015-07-13	[9, 10, 10, 10, 11]	[3, 8, 11, 8, 10]	0.65	false	true