SIMULATION OF VISUAL ASSESSMENT FOR THE GIVEN DEPLOYMENT OF GRAPHICAL USER INTERFACE ELEMENTS

Daniel Skiera
Mark Hoenig
Juergen Hoetzel
Pawel Dabrowski*
Bosch Thermotechnik GmbH
Thermotechnology
Werk Lollar
Postfach 11 61
35453 Lollar, GERMANY
E-mail: Daniel.Skiera@de.bosch.com
E-mail: Mark.Hoenig@de.bosch.com
E-mail: Juergen.Hoetzel@de.bosch.com
E-mail: Pawel.Dabrowski@de.bosch.com

Slawomir Nikiel
Institute of Control and
Computation Engineering
University of Zielona Gora
Podgorna 50
65-246 Zielona Gora, POLAND
E-mail: S.Nikiel@issi.uz.zgora.pl

*Institute of Computer
Engineering and Electronics
University of Zielona Gora
Podgorna 50
65-246 Zielona Gora, POLAND
E-mail: P.Dabrowski@weit.uz.zgora.pl

KEYWORDS

Simulation, Graphical User Interface (GUI), Information Design, Algorithm

ABSTRACT

We observe a constant growth of methods and models enabling automated generation of graphical data, derived either from physical world or taken from simulations. Good information design is crucial for human-computer interface, improving productivity and enhancing human understanding. This paper proposes two novel algorithms to simulate the visual assessment of deployment of GUI elements. GUI elements are treated as the uniform rectangular blocks on the layout. Those blocks eventually are replaced by the system dependent object representations, e.g. visual metaphors of the heating system. The results returned by the algorithms were compared with results obtained by a survey. The proposed metrics can be used in visual evaluation of various simulation models, as long as they consist of logically separable elements. The paper discusses the theoretical background, the properties of proposed algorithms alongside with the sample application prototype outputs.

INTRODUCTION

Good information design requires an understanding of many things. Included are the perception of people: how we see, understand, and think (Winograd and Flores, 1986). It also includes how information must be displayed to enhance human acceptance and comprehension (Miller, 1955; Bodker, 1991). Good design must also consider the capabilities and limitations of the hardware and software of the human-computer interface (Wood, 1997; Teo et al., 2000). Information design, crucial for human-computer interaction (HCI) blends the results of visual design research, knowledge concerning people, knowledge about the hardware and software capabilities of the interfaces and artificial intelligence algorithms. Visual information may be obtained as a result of real-life data collection or as an output of numerical simulations. During the design process the individual elements are assigned visual metaphors. A logical flow of information needs also to be determined.

The next step is to organize and lay out individual elements clearly and meaningfully. Proper screen presentation and structure will encourage quick and correct information comprehension, the fastest possible execution of tasks and functions, and enhanced user acceptance (Gromke, 2007; Hoffmann et al., 2011). It is much harder to estimate numerically how a screen is organized and how its information is actually understood by user, there are some attempts to provide useful metrics (Gamberini et al., 2011). This paper will present the metrics for the redesigned layouts, and the rationale and reasoning that explains why they are useful in automated deployment of visual elements.

Proceedings 26th European Conference on Modelling and Simulation ©ECMS Klaus G. Troitzsch, Michael Möhring, Ulf Lotzmann (Editors)
ISBN: 978-0-9564944-4-3 / ISBN: 978-0-9564944-5-0 (CD)

LAYOUT ALIGNEMENT

Many researchers addressed interface and screen design from the users perspective, giving away hundreds of guidelines for good design in a clear and concise manner (Hartswood and Procter, 2000; Keppel and Wickens, 2004; Galitz, 2007). Generally fewer screen alignments reduce a screens complexity and make it more visually appealing, it is known as the rule of minimum design (Sirlin, 2009). Aligning elements will also make eye movement through the screen much more obvious and reduce the distance it must travel. Screen organization will also be more consistent and predictable. Alignment is achieved by creating vertical columns and horizontal rows of screen fields. Screen balance should be attained as much as possible. But how can we check whether the pattern created is consistent, predictable, and distinct? One of the solutions is the visual complexity measure related to the estimation of the even deployment layout of elements described in the following sections.

ESTIMATION OF EVEN DEPLOYMENT

The estimation of the even deployment can be used for selecting the best even deployments of GUI elements or for construction of a cost (objective) function for a deployment algorithm (Nikiel and Dabrowski, 2011). An example of GUI elements deployment can be seen in the Figure 1.

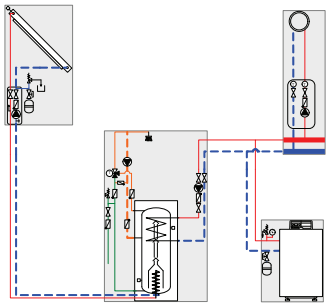

Figure 1: The GUI elements on example of heating system elements.

For the estimation of even deployment algorithms a simple representation of the GUI elements will be used. The elements will be replaced by the rectangles. The rectangles are shown in the Figure 1 (the grey rectangles). The connections between the GUI elements will not be considered in this paper.

Preconditions

Preconditions for the presented estimation of even deployment algorithm:

- the deployment area influences the return results,
- the GUI elements may not overlap,
- the GUI elements should be positioned vertically or horizontally on the given area,
- it's possible to compare the same GUI's with different GUI's where only the deployment of GUI elements is different (the deployment area, the amount and kind of GUI element should remain the same).

The Distances Algorithm

In this algorithm the elements are deployed on the grid (for example one pixel can be treated as a single cell). All distances (vertical and horizontal) between the element and another element or the edges are considered. Figure 2 illustrates the concept of vertical distances for given elements.

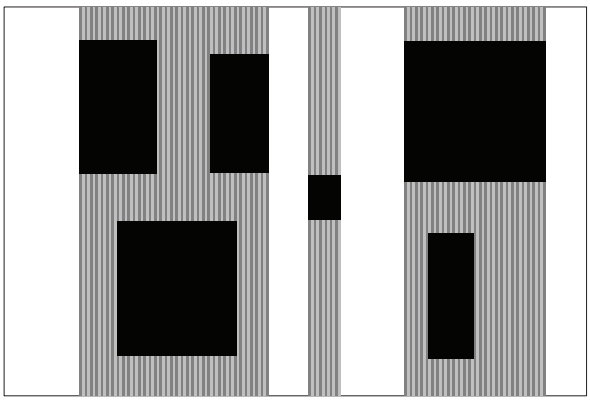

Figure 2: The elements with vertical distances (the length of the gray lines - vertical distances and the horizontal distances are considered).

The idea of this algorithm is based on the fact, that if all the distances have the same length, then the perfect even deployment of the GUI elements is obtained. A histogram of distances is presented in the Figure 3. This histogram was created for the example from Figure 2. The distances have to be determined in the vertical and horizontal direction.

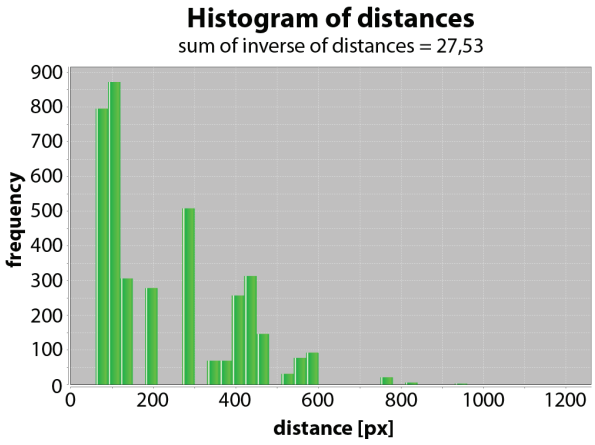

Figure 3: The histogram of distances for the example from Figure 2. The Figure 2 has dimension 1200x800 [px] and the grid cell is 1px.

The sum of squares of distances and the variance of distances were the first metrics used to determine the correctness of even deployment. During our research we compared the returned results by the algorithm for different deployments of the GUI elements on the given area. The smaller the return value of the algorithm, the better the given solution should be. It turned out that these measures classify well most of the deployment examples, but are not resistant to certain cases. En example is shown in the Figure 4.

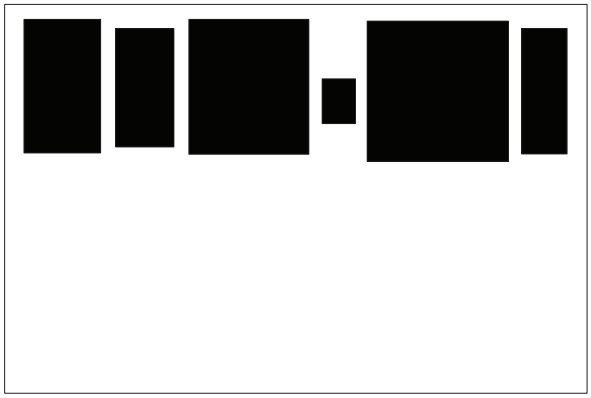

Figure 4: The uneven deployment of GUI elements.

In the situation of grouping of GUI elements the algorithm, which uses the sum of squares of distances or the variance of distances returns good results, despite the fact, that the given deployment is uneven in relation to the area in which the GUI elements have been deployed. This is because of the fact, that the small distances between the GUI elements and the edges are beginning to dominate and they affect the returned result of the algorithm greatly. Figure 5 shows a histogram with the distances for the example from Figure 4. The histogram shows that the short distances dominate.

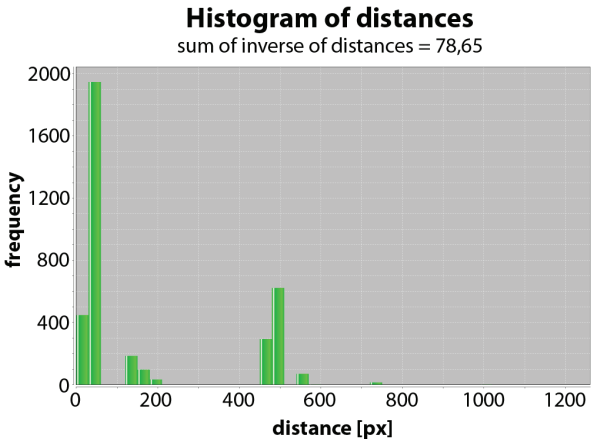

Figure 5: The histogram of distances for the example from Figure 4. The figure 4 has dimension 1200x800 [px] and the grid cell is 1px.

The sum of squares of distances favors short distances between the GUI elements and the edges.

When the elements are grouped, the short distances dominate and the sum of squares of distances is small. This fact should mean that the given deployment is good, but it is not. The solution to this problem would be the sum of the distances raised to more power. This would make the long distances more important than the short distances. However determination of the power is problematic, because it depends on the ratio of the long distances to the short distances. The ratio depends on the considered case.

A similar situation occurs in the variance of distances. If the short distances dominate, the value of the variance is small, because fewer long distances affect the return value non-significantly.

To prevent the above described cases, we should calculate the correctness of the deployment in such a way, that the long distances have to be favored and the short distances have to be "punished". In this case the estimation of simulated deployment of GUI elements have to be factual.

Based on the previous considerations a new approach was created, that sums the inverse of distances between the GUI elements and the edges. This methodology leads to favor the long distances and "punishes" the short distances. The returned value by the algorithm, which uses the sum of inverse of distances is smaller, the deployment of GUI elements on the given area is more even.

After testing, it turned out that this approach works well for estimation of the different configurations of deployments and is fast in cases, in which short distances dominate.

Finally to calculate the even deployment of GUI elements in a given area for the Distances Algorithm the sum of inverse of distances between the GUI elements and the edges is used (see Equation 1).

$$D = \sum_{i=1}^{n} \frac{1}{d_i} \tag{1}$$

where:
D - the sum of inverse of distances
n - the amount of distances between the rectangles and between the edges and rectangles
$i \in N \setminus \{0\}$
d_i - the length of i distance

In the Figure 3 and 5 you can compare the value of the sum of inverse of distances for the two cases.

The time complexity for the Distances Algorithm is equal to $\Theta(N)$.

The Energy Algorithm

The Energy Algorithm is based on the field potential. Some examples we can find in physics e.g.: the Newton's law of universal gravitation (Serway and Jewett, 2010)

or the Coulomb's law (Anaxos Inc. et al., 2009) (see the Equation 2).

$$F = \alpha \frac{o_1 o_2}{r^2} \tag{2}$$

where:
F - the force between the two objects
α - the constant (e.g.: gravitational or proportionality)
o_1 - the first object (e.g.: mass or charge)
o_2 - the second object (e.g.: mass or charge)
r - the distance between the centers of the two objects

For the physical models the objects are treated as a point with mass or charge. This means that for the model the objects do not have physical size and the dimension of the objects is not considered (Anaxos Inc. et al., 2009).

For the Energy Algorithm the constant can ignored, because for our algorithm the constant is not needed. The Energy Algorithm is based on the physical model, therefore we can observe the analogy presented in Table 1.

Table 1: Analogy to physical model

PHYSICAL MODEL	ENERGY ALGORITHM
force	energy
objects (with mass or charge)	GUI elements (width area)
distance between centers of masses or charges	distance between centers of GUI elements

The Energy Algorithm utilizes changes of the value of the power for the distance between centers of two GUI elements. The experiments have shown better results returned by the Energy Algorithm if the power was equal to 4. When the value of power was increased the short distances between the centers of GUI elements were "punished" in greater extent. The given deployment was better, when the associated energy was smaller. Finally the energy between two GUI elements is equal to the Equation 3.

$$E = \frac{a_1 a_2}{r^4} \tag{3}$$

where:
E - the energy between the centers of the GUI elements
a_1 - the area of the first GUI element
a_2 - the area of the second GUI element
r - the distance between the centers of the two GUI elements

The energy for the all given elements is calculated from the Equation 4:

$$E_d = E_e + E_m \tag{4}$$

where:
E_d - the energy for the given deployment of GUI elements
E_e - the energy to another elements, it is calculated from the formula 5
E_m - the energy to mirror elements, it is calculated from the formula 6. The mirror elements are shown in the Figure 6 for one GUI element.

$$E_e = \sum_{0<i<j}^{n} \frac{a_i a_j}{r_{ij}^4} \tag{5}$$

where:
n - the amount of GUI elements, $n \in N \setminus \{0\}$
$i, j \in N \setminus \{0\} \cap 0 < i < j$
a_i - the area of the i GUI element
a_j - the area of the j GUI element
r_{ij} - the distance between the centers of the GUI elements

$$E_m = \frac{1}{16} \sum_{i=1}^{n \in N} a_i^2 \left(\frac{1}{r_{li}^4} + \frac{1}{r_{ti}^4} + \frac{1}{r_{ri}^4} + \frac{1}{r_{bi}^4}\right) \tag{6}$$

where:
$i \in N \setminus \{0\}$
$r_{li}, r_{ti}, r_{ri}, r_{bi}$ - the distance between the center of element and the left, top, right, bottom edge of the area

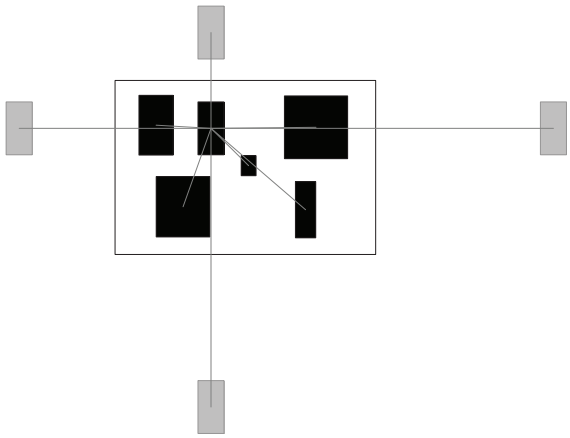

Figure 6: The GUI element with all distances (the gray lines) to another GUI elements and to mirror elements (the gray rectangles).

The mirror elements are used to keep suitable distances of the GUI elements to the edges of the deployment area.

The energy is dependent on the area of the rectangles with GUI elements. The estimation will be better, if the bigger rectangles will be located in longer distances from each other.

The Energy Algorithm unfortunately has some drawbacks, because the approach does not take into consideration the geometry of rectangles. A situation can occur

that for the same GUI elements, but only rotated, the energy will be the same or the distance between rectangles will be to small (see the Figure 7). This can happen, because the Energy Algorithm does not treat the GUI elements as rectangles but as points of mass.

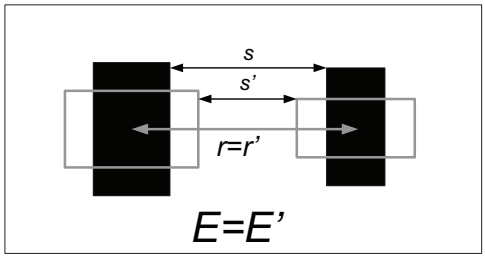

Figure 7: The same energy for two configuration (vertical and horizontal) of GUI elements.

The value of mass is equal to the area of the rectangle and can be treated as circle. The circle has the same area as the rectangle and the point of mass (see the Figure 8).

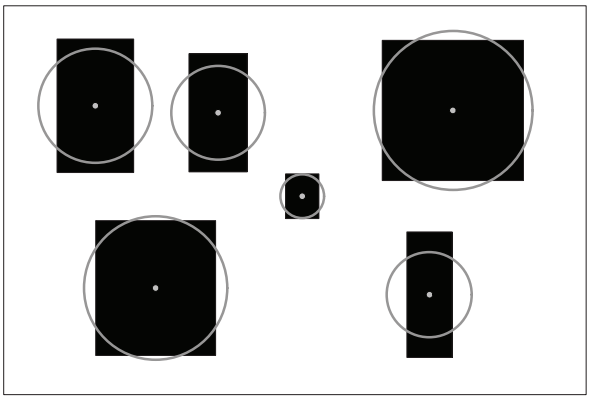

Figure 8: The Energy Algorithm considers the GUI elements as points of mass.

To minimize the above drawbacks we need to make some additional limitations for the Energy Algorithm:

- vertical or horizontal position of the GUI element should not be changed,
- the GUI elements should be represented by the square (in this case the rectangles have the biggest overlap with the circles).

The time complexity for the Energy Algorithm is equal to $\Theta(N^2)$.

Discussion

The disadvantage of the Energy Algorithm is that, it ignores the shape of the GUI elements. With a large amount of elements running time of the Energy Algorithm will rise dramatically, because its time complexity is equal to $\Theta(N^2)$.

The Distances Algorithm allows for the shape of the GUI elements and its time complexity is equal to $\Theta(N)$.

VERIFICATION

In an aim to verify the algorithms sample deployments of the GUI elements were created, which were sorted from good to bad by a group of fourteen people. The respondents received fourteen possibilities for deployment of GUI elements. The elements in all examples were the same. The deployments were printed on cards (an example of a deployment is shown in Figure 4). The cards are shuffled and the respondents sorted the cards from the best to the worst deployment of GUI elements. From the survey results were determined the order of the fourteen deployments. To compare the correctness of simulation of visual assessment the results returned by the described algorithm are presented in a chart. The chart in Figure 9 shows the results obtained from the survey to the results returned by the algorithms.

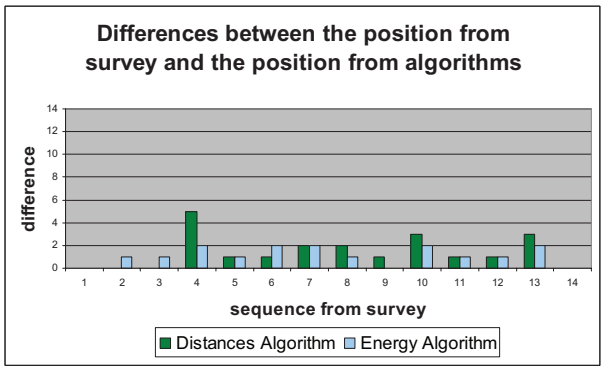

Figure 9: Results from the survey to the results returned by the algorithms.

For better understanding of the chart an example is presented:
For the ninth deployment (the x axis) the Energy Algorithm returned the same position as the position from survey (the difference (the y axis) is equal to zero). The Distances Algorithm returned one more or less position as the position from the survey (the difference is equal to one). This means that the algorithm returned position eight or ten for the deployment of GUI elements). In the best case the returned results by the algorithms should have the difference equal to zero. In this case the simulated visual assessment would be ideal.

As shown in the chart in Figure 9, both algorithms classify the best and the worst deployments of GUI elements in the same place. The differences occur in order of worse deployments, but they are relatively low.

The described algorithms simulate the visual assessment for the given deployment of GUI elements well.

CONCLUSION

Information design delivers constant challenges in the field of HCI. The paper proposes two novel algorithms for estimation of deployment of the visual elements in the given layout. GUI is treated as the output of simulation placing uniform rectangular blocks on the selected

area. The paper proposes alternative metrics of visual complexity, based on the Distances and the Energy techniques. The novel schemes utilize estimation of distances on the image grid to obtain a global goal - the optimization of layout. It is possible to greatly improve automated presentation of data. The authors plan to develop further the method in order to support more complex systems.

ACKNOWLEDGMENT

This study is supported by Bosch Thermotechnik GmbH, and the authors would like to thank the company for the cooperation and assistance rendered.

REFERENCES

Anaxos Inc., Brazell, B., Heckert, P., Nittler, J., Vannette, M., and Willis, M. (2009). *AP Physics B & C 2009*, chapter Coulomb's Law; Field and Potential of Point Charges, page 245. Kaplan Publishing, New York, USA.

Bodker, S. (1991). *Through the interface: a human activity approach to user interface design*. Lawrence Erlbaum Associates, Hillsdale.

Galitz, W. (2007). *The Essential Guide to User Interface Design*. Wiley.

Gamberini, L., Spagnolli, A., Prontu, L., Furlan, S., Martino, F., Solaz, B., Alcaniz, M., and Lozano, J. (2011). How natural is a natural interface? an evaluation procedure based on action breakdowns. *Personal and Ubiquitous Computing*, 5.

Gromke, G. (2007). Digital asset management - der effektive umgang mit mediendaten. *Proceedings, Intl. Conf. EVA 2007 Berlin*, pages 161–166.

Hartswood, M. and Procter, R. (2000). Design guidelines for dealing with breakdowns and repairs in collaborative work settings. *International Journal on Human Computer Studies*, 53:91–120.

Hoffmann, P., Lawo, M., and Kalkbrenner, G. (2011). Zur aesthetik interaktiver medien - hypervideo im spannungsfeld zwischen usability und design. *Proceedings, Intl. Conf. EVA 2007 Berlin*, pages 117–123.

Keppel, G. and Wickens, T. (2004). *Design and analysis: a researchers handbook*. Pearson/Prentice Hall, Upper Saddle River.

Miller, G. (1955). The magical number seven, plus or minus two - some limits on our capacity for processing information. *In: Psychological Review*, 101(2):343–352.

Nikiel, S. and Dabrowski, P. (2011). Deployment algorithm using simulated annealing. In *Methods and Models in Automation and Robotics - MMAR 2011 : 16th international conference*, pages 111–115, Miedzyzdroje, Poland.

Serway, R. and Jewett, J. (2010). *Physics for Scientists and Engineers*, volume 1, chapter Newton's Law of Universal Gravitation, page 375. Brooks Cole, Belmont, USA, 8 edition.

Sirlin, D. (2009). Subtractive design. *Game Developer*, pages 23–28.

Teo, L., Byrne, J., and Ngo, D. (2000). A method for determining the properties of multi-screen interfaces. *International Journal of Applied Mathematics and Computer Science*, 10(2):413–427.

Winograd, T. and Flores, F. (1986). *Understanding computers and cognition*. Ablex Publishing, Norwood.

Wood, L. (1997). *User Interface Design: Bridging the Gap from User Requirements to Design*. CRC Press.

AUTHOR BIOGRAPHIES

DANIEL SKIERA is currently working as a Software Architect at Bosch Thermotechnik GmbH, Lollar, Germany. He obtained a PhD in Physics at the University of Giessen, Germany. His scope of work included data analysis, image processing and system simulation. His e-mail is Daniel.Skiera@de.bosch.com.

MARK HOENIG is currently working as a group manager at Bosch Thermotechnik GmbH, Lollar, Germany. He graduated in Physics at the University of Goettingen and received a PhD from the University of Cologne. His work included data analysis and nonlinear optimization. His e-mail is Mark.Hoenig@de.bosch.com.

JUERGEN HOETZEL is currently working as manager at Bosch Thermotechnik GmbH, Lollar, Germany. He graduated in Electronics at the University of Darmstadt and received a PhD from the Technical University Berlin. His work included system architecture and Internet connectivity. His e-mail is Juergen.Hoetzel@de.bosch.com.

SLAWOMIR NIKIEL is currently the Professor at the Institute of Control and Computation Engineering, Department of Electrical Technology, Computer Science and Telecommunication, University Of Zielona Gora, Poland. His research interest include HCI, game programming and multimedia systems. His e-mail is S.Nikiel@issi.uz.zgora.pl.

PAWEL DABROWSKI is currently the PhD student at the Institute of Computer Engineering and Electronics, Faculty of Electrical Engineering, Computer Science and Telecommunications, University Of Zielona Gora, Poland. His dissertation is developing in cooperation with Bosch Thermotechnik GmbH, Lollar, Germany. His research interest include automatic deployment of GUI elements. His e-mail is P.Dabrowski@weit.uz.zgora.pl or Pawel.Dabrowski@de.bosch.com.