Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 131 (2015) 71 – 85

Procedia Engineering

www.elsevier.com/locate/procedia

World Conference: TRIZ FUTURE, TF 2011-2014

Exploiting TRIZ tools in Interaction Design

Stefano Filippi[a], Daniela Barattin[a]

[a] *University of Udine, DIEGM Dept., Via Delle Scienze, 208, 33100, Udine, Italy*

Abstract

Current market laws ask for new product development methods and tools dealing with both technological and interaction related issues. Starting from this, the research described in this paper aims at finding and exploiting some TRIZ tools in interaction design field of practice. TRIZ theory offers well defined and structured methods and tools, and allows generation of generic guidelines for innovative design of a huge variety of products. Interaction design focuses on studying and developing a correct interaction between users and products, in order to maximize cognitive compatibility. Real goal of this research is developing a new design method where systematic approach to innovation of TRIZ compensates for some lacks of the user centered interaction design process. Starting from similarities and differences between tools currently used in these two domains, the research considers the TRIZ thirty-nine features, the forty inventive principles, and the contradiction matrix. These elements are adapted to the requirements of the interaction design field of practice, obeying to classic usability rules.

These new elements contribute to the definition and development of a new design framework named interaction design guidelines - IDGL. The effectiveness of these elements has started to be tested in a case study focused on the interaction design of a new DVD recorder.

Keywords: Interaction Design, TRIZ, Inventive Principles, TRIZ Contradiction Matrix, TRIZ Features;

1. Introduction

Nowadays, industries obey to fast and continuously evolving market laws asking them for an always shorter time to market, sometimes with scarce resources. In the meantime, customers want innovative products. These must be cost effective, embedding state of the art technology, portable and multifunctional, customizable and easy to use [1]. All of this implies a revision of classic design methods, because focusing on functionalities and technological issues is not enough anymore. Real users' needs must be accounted for, otherwise products

doi:10.1016/j.proeng.2015.12.349

could be refused, even if correctly designed and developed [2]. Soft reliability is the name of a special branch of interaction design specifically addressing this topic [3].

Then, new methods and tools are required, in order to consider both technological and interaction aspects during product design. All of this suggests looking for a possible synergy between systematic approach to innovation of TRIZ theory and current interaction design procedures. The ultimate goal would be to enrich both domains, by adding some systematic nature to interaction design methods, and enlarging the application coverage of TRIZ theory.

After this introduction, paper runs as follows. First, state of the art of TRIZ and interaction design is summarized. After that, a possible interaction/integration/synergy between them is considered, and some new design tools are presented. Then, a case study is performed, as first stage of testing and validating the research results. Finally, some discussion deals with possible improvements that can occur in the two domains, and paper ends with some hints about future research directions.

2. Tools and methods

This paragraph summarizes meaningful features of the two domains considered in this research: TRIZ theory (with its evolutions) and interaction design.

1.1. TRIZ theory

TRIZ is the Russian acronym for the inventive problem solving theory, developed by G. Altshuller starting from the fifties [4]. It aims at generalizing and systematizing the design process, in order to boost the generation of innovative ideas, concepts, and products.

Modern design problems are more complex than when TRIZ was born. Today more domains are involved, and focusing on technological issues is not enough anymore. Moreover, classic TRIZ manages one problem at a time, while today the requirement is in dealing with many of them contemporarily.

For all these reasons TRIZ has evolved towards a more articulated theory named OTSM. This is the Russian acronym for general theory of powerful thinking [5,6,7,8]. OTSM revises the bases of TRIZ in a formal way, while keeping contradictions as central element.

Now, contradictions describing a complex problem are ordered and related each other in a network. At the beginning, the network of problems relates sub-problems and partial solutions. Then, technical contradictions are extracted and related each other in the network of contradictions. Finally, parameters that rule the relationships between contradictions are collected in the network of parameters. This representation of the complex problem is the starting point for the generation of design solutions [5,7]. This activity exploits classic TRIZ tools as ariz, contradiction matrix and trends of evolution. They are conveniently modified to manage more than one contradiction simultaneously.

Research presented in this paper mainly focuses on the feasibility of TRIZ exploitation in interaction design field of practice, so only classic release of three TRIZ tools has been considered at the moment: forty inventive principles, contradiction matrix and thirty-nine features.

Forty inventive principles are one of the results of Altshuller's patent mining activities [9,10]. The definition of these principles is general enough to be effectively exploited in many different domains, for solving many different types of design problems. These principles help a lot in clearing psychological inertia, since they allow exploiting results coming from domains sometimes very far from designers' ones. Together with the forty principles, another element of interest here is the contradiction matrix [10,11].

When two engineering requirements - named features in TRIZ theory – generate a contradiction, the entry of the contradiction matrix corresponding to row and column of the two requirements suggests the best inventive principles that could help in solving the contradiction.

TRIZ features are thirty-nine, and they allow the physical characterization of product under development [11,12].

In the last decade TRIZ has become one of the fundamentals for systematic and innovative design. Some software packages are on the market, directly implementing classic TRIZ theory or belonging to several specific design domains but in any case referring to it. Among them, there is Goldfire developed by Invention Machine [13], TRISolver by TriS Europe GmbH [14], IWB - Innovation WorkBench by Ideation International [15], and Triptych by the Statistical Design Institute [16]. On the contrary, OTSM is less present in commercial software packages; up to now it has been implemented only in research prototypes.

2.1. Interaction Design

Interaction design - ID - focuses on correct interpretation and implementation of user-product dialogue. It allows generating products ready to be easily and intuitively used since the beginning, and accepted by the most of the users, thus avoiding soft reliability problems [3]. It is quite young as a research field, and full of methods and tools based on the usability concept. As the ISO 9241 standard says, usability is "the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments" [17]. Specifically, ID methods and tools focus on user satisfaction because new definitions of product quality are heavily based on it. Unfortunately designers' conceptualization of possible users rarely matches with reality; this is why more than often real users are directly involved in design activities. Literature describes existing ID approaches focused on generalizing specific users' needs, complaints, and problems, highlighted before and during the use of specific products. All of this in order to generate an actual starting point for the generation of interaction design guidelines. Among recent approaches, DCART [18] is the software implementation of a procedure that translates raw input data into usability problems, in order to highlight the correct ones, mainly by not so skilled usability practitioners. This translation comes through intermediate stages, named usability problem instances - UPI. They are generated in an automatic way, and aim at recognizing, classifying, and generalizing users' requirements.

LINK-UP [19] has the same goals as DCART, but follows a different procedure. First, interaction designers formally describe the product under development. Starting from this description, LINK-UP builds scenarios and suggests task models, aiming at understanding interaction requirements as best as possible. Task models are collected in a database and could be reused in similar situations. Once again, LINK-UP helps usability experts in correctly identifying interaction requirements.

Inner complexity of these two approaches seems to be quite high, so their adoption is not straightforward. Moreover, when the product under development is completely different from the ones analyzed before, too much knowledge about their mechanisms and structure seems needed in order to use them properly. Finally, none of them deal with the generation of some sort of explicit, ready-to-use interaction design guidelines. This is one of the real challenges, because guideline generation is mainly a creative and subjective task, and for this reason very difficult to deal with in an analytic way.

Interdisciplinary approach with a user-system focus described in [3,20] is more complete. It collects input data and elaborates them towards the generation of real design solutions. Collected data regard product usage and users' behavior and impressions during it. Collecting activities come thanks to a software package running in the product itself, the dynamic product usage information system - D'PUIS. Process mining - ProM - techniques allow analyzing these data, while the compromise decision-support- problem - cDSP - tool is devoted to the generation of interaction design guidelines [21]. cDSP allows associating the right information to the right stakeholder (interaction designer, quality expert, marketing agent, etc.) at the right time. The main difficulty in adopting this design approach seems to be the setup and use of D'PUIS software package.

Another important help for interaction design comes from eco-sustainability. Energy waste can be heavily reduced thanks to suggestions about correct use of products. These suggestions are translated into design rules in order to generate products that best support energy saving, thanks to the optimization of their user interaction. Eco-feedback theory [22,23] suggests the correct behavior to the users, thanks to messages written on labels. Scripting principle [22,23] tries to maximize affordance, in order to make correct actions available and to prevent unwanted ones. These approaches give only suggestions at the moment; they don't appear as structured as the previous ones, and for this reason they cannot be exploited in interaction design as they are.

Finally, some usability rules are of interest here, mainly because correctness of TRIZ tool customization for interaction design has been controlled using them. They are seven dialogic ISO principles [24], eight Shneiderman's golden rules [24], ten Nielsen's heuristics and interaction paradigms of Dix, Finlay, Abowd e Beale [25].

3. Activities

This paragraph describes the definition of three tools for interaction design derived from TRIZ theory. These tools will contribute to the generation of a complete interaction design framework named IDGL - interaction design guidelines.

3.1. Analogies and differences between TRIZ and ID

Highlighting analogies and differences between TRIZ and ID is important, because it allows finding those contact points that will be exploited in the following. Table 1 summarizes meaningful analogies.

Inventive principles can be put into relationship with laws of interaction design, Nielsen's heuristics, and Norman's design principles, in terms of design guidelines that can be generally used for finding the best solutions of a design problem. Besides, these guidelines can be applied in different contexts and for different kinds of problems. There are also some correspondences between TRIZ laws of system evolution and interaction paradigms and metaphors. When the interaction designers imagine future developments of the interaction of a product, they set a sketch of technology forecasting enforced by the use of metaphors. Another correspondence can be identified between TRIZ Ideal Final Result and usability evaluation and testing methods. This correspondence is intended as the will of interaction designers to develop products as usable as possible or, in other words, ideally usable. Moreover, TRIZ functionality and trimming may be related to ID metaphors. In fact, interaction designers set some sort of functional representation of the system interface during the definition of metaphors. This representation is not as structured as a TRIZ system functional model; anyway it is an interesting contact point between these two domains. Finally, an analogy can be identified between TRIZ multi screen approach and the ID metaphors and interaction paradigms. These two ID items allow designers analyzing the product regarding its status in time (present, past and future) and space (itself, subsystems/subfunctions, environment), exactly as it happens thanks to the nine box approach.

Table 1. Analogies between TRIZ and ID

TRIZ	ID
Inventive Principles	Laws of interaction design Nielsen's Heuristics Norman's design principles
Laws of system evolution	Interaction paradigms Metaphors
Ideal Final Result	Usability Evaluation&Testing methods
Functionality and trimming	Metaphors
Multi Screen or Nine box	Metaphors
Approach	Interaction paradigms

Together with the analogies shown in table 1, it could be useful to highlight main differences between the two domains as well, because the search of synergies may effectively start from them, in trying to compensate lacks of ID with peculiarities of TRIZ, and vice versa. For example, a close correlation between TRIZ contradictions and any ID item was not found. We have only found a soft correlation between TRIZ contradictions and Nielsen's heuristics. Every time a usability heuristic is violated in the interface of a product under development, a sort of behavioral contradiction may be highlighted.

Table 2. Main differences between TRIZ and ID

TRIZ	ID
Highly structured approach	Loosely structured approaches
Focus on functionality and technical issues	Focus on interaction aspects and users' needs
Emphasizes abstraction	Emphasize the real context
Describes 'what' and 'how'	Describe 'why'

As summarized in table 2, main differences between these two domains can be identified in the following aspects. First of all, there is a highly structured approach to problem solving in TRIZ theory, against a loosely structured problem solving strategy in ID. Absence of a systematic design approach is highlighted by the different aims. TRIZ focuses on functionality and technical aspects of problems, while ID is oriented to interaction and user's needs. Moreover, TRIZ theory and concepts emphasize abstraction, while ID methods emphasize real contexts. Finally, in ID practices innovation is intended as a consequence of skill and experience of the design team members. For this reason, results of ID reasoning refer to description of the 'why' aspects of design process, but often all of this does not suggest solutions. On the other hand, TRIZ, thanks to its structured approach and leveraging of other successful results, may suggest 'what' and 'how' aspects of design problems and, in this way, it can prescribe real solutions.

3.2. Development of new tools

New interaction design tools can be created now, thanks to analogies and differences highlighted in previous paragraph. The IDGL framework is introduced first, because it is the place where new tools are exploited. Then, the three TRIZ-derived tools are described in detail.

IDGL overview

IDGL is an interaction design framework where innovative interactive products can be developed in an effective way, thanks to a set of guidelines generated semi-automatically. It is composed by several modules linked each other and working in synergy, and its architecture is based on classic design structures, as the house of quality coming from quality function deployment theory [26].

IDGL adoption has well defined steps. First, features of the product under development are described, so as so the final users' characteristics. Then IDGL generates a questionnaire in order to collect and refine users' needs and expectations. All these pieces of information go to fill the house of interaction, the main data structure of the IDGL derived from the house of quality. Thanks to it, interaction requirements are highlighted, and design guidelines are generated accordingly. In doing so, other knowledge is exploited, derived directly from the classic TRIZ components described in tools and methods section, modified and customized according to interaction design requirements.

Interaction principles

Customization of TRIZ tools starts with the generation of interaction principles. The forty principles of TRIZ theory have been used as source for this. As TRIZ principles come from the analysis of engineering patents, they strictly refer to that domain. For this reason, they have had to be translated into something closer to interaction issues. In doing so, classic usability rules acted as a check to evaluate congruity and validity of the result. The experience in the field of some usability experts has been exploited as well. They have been involved because it would have been impossible to replicate the huge work that Altshuller performed in the technological field, given current time, material and available resources in general. The same consideration is worth for definition and validation of the other tools described hereafter.

The result consists in a list of forty-seven interaction principles. Each principle has some examples of application, to make its comprehension and adoption easier. The resulting amount of principles is not equal to forty because some TRIZ principles generated more than one interaction principle, while not all TRIZ principles were suited for the generation of corresponding interaction principles. Complete list of principles is in appendix.

For example, fifth TRIZ principle "Merging" suggests to "Bring closer together (or merge) identical or similar objects, assemble identical or similar parts to perform parallel operations" [11]. In the interaction design domain, this principle becomes "Combine more elements in order to generate forced paths avoiding multiple selections". One of the related examples sounds as "Programming a VHS recorder, allow the use of a single numerical code (e.g. the ShowView), instead of demanding manual input of recording date, time, program, etc.".

Interaction principles have been classified into four different categories, using a two-dimensional grid. First dimension refers to physical vs. psychological aspects of design or, in other words, to interface issues vs. cognitive compatibility and problem solving; second dimension tells if principles aim at increasing vs. decreasing some interaction aspects. The same interaction principle could fall in more than one category. Regarding the previous example, the "Merging" interaction principle is categorized as physical/decreasing. In fact, it aims at lowering the number of actions during the problem solving process. This categorization is important because it allows mapping interaction principles with other pieces of information that will be defined in the next paragraphs.

Interaction requirements

Once defined the principles that could suggest hints for generating interaction design solution concepts, now there is the need to develop elements where to build these concepts on. Their role consists in making usability evaluation as objective as possible, by providing a set of quantitative indices and minimizing the influence of personal judgments. Moreover, they should boost a homogeneous management of different classes of products, by representing interaction aspects in a general way. For this reason, they are defined a priori, and their list is one more piece of information in the IDGL knowledge base.

Up to now, thirty-one interaction requirements have been highlighted and categorized. They come from the analysis of many projects where interaction design had a major role, and from the suggestions of thirty-nine TRIZ features. They are quite different from these features, because information content is richer and more articulated. As it happens for engineering requirements in the house of quality, each interaction requirement is numbered, has a title, an improvement direction, a unit of measurement, and a category. This categorization is exactly the same used for interaction principles described before. Appendix reports full list of the interaction requirements.

An example of interaction requirement is "Universal symbols". Its improvement direction is "up" because as the number of universal symbols rises, users interact more easily with the product by exploiting their own experience. The unit of measurement is "#" and the category is "physical/psychological/increasing", because it regards both the psychological and the physical aspects of interaction and refers to the growth of universal symbol amount. Another example is "Data needed to be memorized to perform actions", direction is "down", unit "#", and category "physical/psychological/decreasing".

Relationship matrix

Any interaction requirement could be accomplished thanks to the help of one or more interaction principles. For example, the requirement "Default settings", representing the amount of chunks of information that the product can supply automatically, could be implemented at best thanks to hints coming from the principle "Merging". In fact, this one encourages grouping data and the resulting groups allow product users a heavy exploitation of default entries because the simple indication of a specific group determines the input of multiple values automatically.

The data structure named relationship matrix has been chosen as the best way to collect relationships between requirements and principles. Rows and columns contain the same list of all interaction requirements, as shown in figure 1.

Entries of the main diagonal contain links to the principles directly related to each requirement. These relationships have been set up thanks to hints coming from the TRIZ theory and with the help of the shared categorization cited in the previous paragraphs.

The other entries of the relationship matrix are as important as main diagonal ones. They refer to relationships between couples of requirements. Analogously with the TRIZ contradiction matrix, where a contradiction expressed by two antagonist engineering requirements could be solved thanks to some suggested TRIZ principles, here the concept has been exploited first, and further developed afterwards.

As it happens in engineering domain, interaction design can highlight contradictions between requirements as well. In this case, IDGL behavior borrows as-it-is from TRIZ; some interaction principles are suggested in order to solve a contradiction. Links to these principles appear in so called "contradictory half entries" of the relationship matrix. For example, consider requirements "Time to perform actions" and "Time for messages before they disappear". First one should decrease, while the second should increase or, in any case, should allow the complete comprehension of message by the product user. Linked principle "Partial or excessive actions (about increase potentiality)" suggests "Set persistence of system messages according to user type. Younger users or skilled ones need shorter time than beginners or elder people".

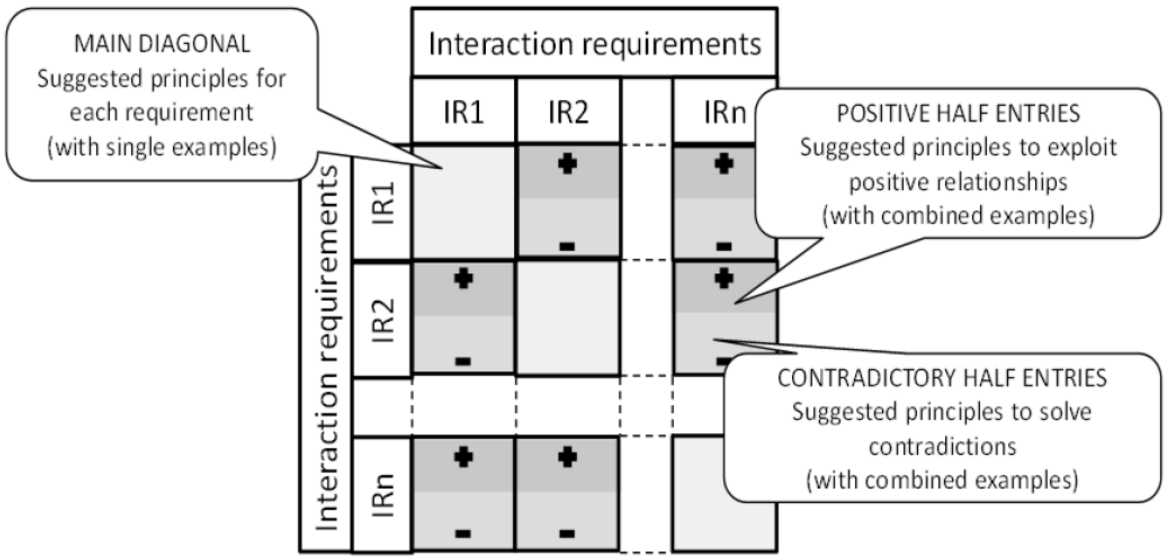
MAIN DIAGONAL
Suggested principles for each requirement (with single examples)
Interaction requirements
IR1
IR2
IRn
Interaction requirements
IR1
IR2
IRn
POSITIVE HALF ENTRIES
Suggested principles to exploit positive relationships (with combined examples)
CONTRADICTORY HALF ENTRIES
Suggested principles to solve contradictions (with combined examples)

Figure 1. The relationship matrix

Regarding the further development, improvement stands in the fact that IDGL considers positive interactions too. In other words, for sure two requirements could obstacle each other, but two requirements could as well contribute to enhance quality of interaction design if considered together, in synergy. For example, requirements "Universal symbols" and "Data visibility" could be positively related, because enhancement of first one makes the second better as well. Suggested interaction principles exist even for positive relationships between requirements, and their links appear in the "positive half entries" of the relationship matrix. Regarding last example, "Partial or excessive actions (about decrease potentiality)" and “Local quality” principles could help, given that they recommend "a) the use of symbols already present in the product application domain, and b) to differentiate symbols enough to avoid misunderstandings".

Analogously with single examples associated to interaction principles, relationships encoded in the half entries of the relationship matrix have "combined examples". They are collected in two different lists, one for the contradictory relationships and the other for positive ones.

The nature of the relationship, contradictory vs. positive, depends from the context where relationships are considered. As an example of double content in the same matrix entry, the two requirements "Self- tuning functions" and "Customizable functions" can be in a contradictory or positive relationship as follows. Contradictory relationship could happen when using, for example, a word processor. The problem is that the customization of the automatic corrector of typos could be very hard to be reached or, even worse, impossible. In this case, interaction principle "Preliminary actions - Add actions that run automatically by forecasting user behavior" is suggested. Thanks to this suggestion, designers can think about a smart algorithm that memorizes the way users interact with the word processor and how they correct their mistakes, and then can customize typo corrector automatically. On the other hand, a positive relationship between the two requirements can be found when dealing with a modern vacuum-cleaner. The user customizes some parameters, like power, and cleaning type and frequency. Meanwhile, the product performs a self-tuning during cleaning to prevent breakdowns. Interaction principle "Anti-weight
- Lower the amount of data to be memorized by users, by exploiting default values and/or tracing of system states", could suggest the development of a fully-automatic vacuum-cleaner, self-moving, able to decide autonomously where, when, and how to clean, without the need of an explicit training performed by the user. This way, the cognitive load lowers, thanks to automatisms that make the system self-tuning given the environmental situation. Of course, some interaction would be still possible, but it is not strictly required as before. An implementation of all of this can be found in modern robot vacuum-cleaners.

The whole relationship matrix is reported in Appendix.

4. Case study

A case study focused on interaction design of a new DVD recorder is conducted as a starting point to test and validate the results of the research. Actually the whole IDGL framework is exploited, but only information related to the new TRIZ-derived tools is reported here for space reasons.

Twenty users were required to answer a questionnaire. They have been classified regarding functionalities they usually use. Five people simply watch DVD movies; six are able to record a TV program while watching it too; five are good at managing the content of the internal hard disk; four, the most skilled ones, are familiar with recording timing too.

Questionnaire contained 17 multiple-choice questions covering many topics, from users' characteristics to problems that they usually face with. Analysis of the answers highlighted 63 interaction aspects that represent users' requirements and expectations about the product under development.

The house of interaction has filled automatically with these aspects and with other pieces of information available in the IDGL knowledge base. The outcomes were 20 interaction requirements meaningful for the DVD recorder. Two of them are "Visibility of interaction components", and "Universal language". They represent the input for exploiting the relationship matrix. First step consisted in looking at the main diagonal. For any requirement it highlights the interaction principles that give single examples to build the list of design guidelines. Then, designers could generate concepts that implement the requirements and this happened thanks to these examples.

All together, 159 new single examples were offered. Two of them, coming from REQ5 - "Universal symbols", are:

- "Use symbols already used in similar products (VCR, CD players, etc.)";
- "Use symbols compatible with user category (young people, elder ones, skilled vs. beginners, etc.)".

Once exploited single examples, relationship matrix was considered again. This time, cells other than the main diagonal were exploited. They contain positive and negative relationships between requirements. Again, each of them contains the interaction principles involved, together with combined examples to be potentially exploited in order to complete the final list of guidelines. Designers could develop more articulated concepts than before, because more requirements are involved.

135 combined examples were offered. For example, starting from two requirements "Feedbacks of system state changes", and "Helping messages during actions needed to perform functions", positive relationship between them suggested the combined example:

"Feedbacks can appear as popup windows containing messages. Attributes of window background (color, transparency, etc.) could be exploited to contextualize messages at best, for example when they refer to some important current or future change of system state (warnings about possible errors, notifications of interrupts that made the system unstable, etc.)".

IDGL collects all the pieces of information described up to now and represents them as reports, where all data related to the product under development are collected. They contain the product features, the house of interaction, the single and the combined examples acting as interaction design guidelines, and many other useful data.

5. Discussion and conclusions

This research started from exploring a possible exploitation of some TRIZ tools in the interaction design field of practice. Analogies and differences between these two disciplines allowed identifying three tools as the starting point of this synergy, that in the end resulted in the development of the IDGL, a complete framework for interaction design. The resulting new items - interaction principles, interaction requirements, and relationship matrix -, could become one of the ways to make TRIZ familiar to a wider community, to associate a more structured approach to interaction design (thanks to the whole IDGL framework too), and to add important engineering contents to the management of usability issues during product development.

TRIZ theory might somehow benefit from this research as well. For example, TRIZ features could be enhanced looking at the richer current format and content of the interaction requirements. In fact, they have units of measurement, trends, a classification, and they focus not only on physical aspects but consider also conceptual issues related to user-product dialogue. Moreover, interaction principles could suggest some extension of the TRIZ vocabulary, in widening the collection of possible application domains. Maybe, relationship matrix is the new tool that could better contribute to some TRIZ enhancement. This matrix synthesizes the set of principles and the contradiction matrix in a unique entity. Moreover, it introduces the new concept of positive relationship, not present in TRIZ.

Results of a first case study seem to validate the structure and content of IDGL and, in particular, they start to witness the effectiveness of the new three TRIZ-related tools.

Some other research is needed, to make IDGL a robust design tool and to widen its coverage to a larger collection of product types. First of all, a quantitative evaluation of IDGL performance would be required. The results of other researchers' work, as the evaluation method proposed by Verhaegen et al. in [27], could be exploited in this sense. Moreover, each IDGL adoption could enrich its knowledge base. In fact, one of the current research topics is the development of a mechanism able to translate solution concepts generated by designers thanks to the IDGL into single and combined traceable examples, to be added to existing ones and made available for next adoptions of the IDGL framework. Finally, the large number of single and combined examples generated by IDGL and offered as guidelines suggests and requires studying how to browse and exploit them in the most usable way.

Appendix A. Interaction principles, interaction requirements and relationship matrix.

Table 3. Interaction principles

#	TRIZ	Name	Meaning in ID	Categories
1	1	Segmentation	Decompose complex actions into simpler ones	Ph/In
2	2	Taking out	Trim functions or components not important from the users' point of view. Focus on those ones that could generate dissatisfaction if missing	Ph/In
3	3	Local quality	Make system behaviors really different each other, in order to be recognized immediately, without misunderstandings	Ph/Ps/In
4	4	Asymmetry	Make components performing different functions very different each other, to enhance recognition easiness	Ph/In
5	5	Merging	Combine more elements in order to generate forced paths avoiding multiple selections	Ph/De
6	6	Universality	Make components in order to perform parallel functions, triggered by simple user actions	Ph/In
7	7	Nested doll	Embed functions one another in order to perform all of them with the same actions as before	Ph/De
8	8	Anti-weight	Lower the amount of data to be memorized by users, by exploiting default values and/or tracing of system states	Ph/Ps/De
9	9	Preliminary anti-action	Insert automatic actions performed by the system to avoid dangerous/error states	Ph/In
10	10	Preliminary action	Add actions that run automatically by forecasting user behavior	Ph/In
11	11	Beforehand cushioning	Provide messages and hints every time a possible error state is going to occur	Ph/In
12	12	Equipotentiality	Limit variations about user skill requirements	Ps/De
13	13	The other way round (actions)	Switch actions each other to lower interaction complexity	Ph/De
14	13	The other way round (components)	Switch components each other to enhance learnability and memorability	Ph/De
15	14	Curvature	Differentiate shapes of interaction components to enhance affordance	Ph/In
16	15	Dynamics (components)	Dynamize shapes and positions of interaction components to enhance usability	Ph/In
17	15	Dynamics (actions)	Dynamize towards "intelligent" systems, able to make decisions autonomously	Ph/In
18	16	Partial or excessive actions (about increase potentiality)	Enhance functional capability according to user skill	Ph/In
19	16	Partial or excessive actions (about decrease potentiality)	Lower functional capability if too much cognitive load is required	Ph/De
20	17	Another dimension (actions)	Avoid the need of messages/helps/hints in performing actions by moving on the affordance dimension	Ph/In
21	17	Another dimension (components)	Move from a physical interaction to a contact-free one	Ph/In
22	18	Mechanical vibration	Exploit vibrations to enhance feedback quality	Ph/In

23	19	Periodic action	Show hints/helps automatically after quite long unused periods	Ph/In
24	20	Continuity of useful action	Enhance system transparency and avoid breaks during problem solving activities	Ph/Ps/In
25	21	Skipping	Exploit prior knowledge about states of dialogue to skip parts of it	Ph/Ps/In
26	22	Harm to benefit (components)	Substitute components showing obsolete shapes that recall different actions/functions	Ph/Ps/De
27	22	Harm to benefit (actions)	Convert wrong actions into knowledge, both for system and users, to prevent them occurring again	Ps/In
28	23	Feedback	Provide correct feedback for each user action, also to avoid modal behavior	Ph/In
29	24	Intermediary (possible actions)	Introduce temporarily external helps to manage error states	Ps/In
30	24	Intermediary (required actions)	Insert intermediate actions to get dialogue easier and faster	Ph/In
31	25	Self-service	Avoid exploitation of external help in performing actions, and improve interface components (colors shapes, positions, etc.) to enhance affordance	Ph/Ps/In
32	26	Copying	Allow trying system interaction before buy it and/or to learn it, thanks to demonstrators	Ph/In
33	27	Cheap short-living objects	Substitute complex actions - they require high skilled users - with "cheaper" - less demanding - ones	Ps/In
34	28	Mechanics substitution	Substitute components that force physical interaction with others allowing contact-free dialogue	Ph/In
35	29	Pneumatics and hydraulics	Simplify physical interaction by substituting solid components with graphical artifacts	Ph/In
36	30	Flexible shells and thin films	Use flexible and/or thin interface components to make interaction lighter	Ph/In
37	32	Color changes	Set colors according to system states	Ph/Ps/In
38	33	Homogeneity	Spread homogeneity all over interface components	Ph/Ps/In
39	34	Discarding and recovering (help)	Hide hints/helps/messages getting obsolete, but keep them ready to be updated in order to become useful again if something happens	Ph/Ps/In
40	34	Discarding and recovering (actions)	Analyze and enhance predictable actions to switch them into successful elements and collect successful actions from different systems to enhance learnability and affordance by exploiting user experience	Ph/In
41	34	Discarding and recovering (components)	Substitute obsolete interaction components with newer once with higher affordance	Ph/In
42	35	Parameter changes	Analyze changes of system state to highlight possible automation in managing system parameters	Ph/In
43	36	Phase transitions	Exploit transitions between dialogue states to enhance feedback	Ph/In
44	37	Thermal expansion	Using some sort of expansion (not necessarily the physical one) to give the user the right help in case of possible error states	Ph/Ps/In
45	38	Strong oxidants	Using external additive (e.g., virtual reality) to enhance affordance and learnability of interaction.	Ph/In
46	39	Inert atmosphere	Condition the workspace in order to let it be always the same during the whole session	Ph/In
47	40	Composite materials	Make dialogue stronger, by putting several dialogue elements and components together	Ph/Ps/In

Table 4. Interaction requirements

#	Name	Units	Impr.	Categories
1	Visibility of interaction components	%	Up	Ph/In
2	Shapes of interaction components	#	Down	Ph/De
3	System dimensions	mm3	Down	Ph/De
4	Feedback of system state changes	#	Up	Ph/In
5	Universal symbols	%	Up	Ph/Ps/In
6	Symbols	#	Down	Ph/De
7	Shape reuse in symbols	#	Down	Ph/De
8	Universal language	%	Up	Ph/Ps/In
9	Language uniformity	%	Up	Ph/Ps/In
10	Abbreviations	#	Down	Ph/De
11	Action visibility (affordance)	%	Up	Ph/In
12	Data visibility	%	Up	Ph/In
13	Parameters able to be changed without breaking the problem solving	#	Up	Ph/In
14	Functions that, once finished, come back home	#	Up	Ph/In
15	Data needed to be memorized to perform actions	#	Down	Ph/Ps/De
16	Needed steps to perform actions	#	Down	Ph/De
17	Default settings	#	Up	Ph/In
18	Actions to preserve important system data	#	Up	Ph/In
19	Self-tuning functions	#	Up	Ps/In
20	Customizable functions	#	Up	Ph/In
21	Interaction components for fast exit from functions	#	Up	Ph/In
22	Error messages	%	Down	Ph/De
23	Helping messages during actions needed to perform functions	#	Up	Ph/In
24	Messages after actions explaining consequences	#	Up	Ph/Ps/In
25	Time to perform actions	sec	Down	Ph/De
26	Time for messages before they disappear	sec	Up	Ph/In
27	Matching between real action sequences and user manual ones	%	Up	Ph/In
28	Matching between real functions and user manual ones	%	Up	Ph/In
29	Missing functions	#	Down	Ps/De
30	Interchangeable interaction components	#	Up	Ph/In
31	Time for substitution of interaction components	sec	Down	Ph/De

Table 5. Relationship matrix

Req.	1	2	3	4	5	6	7	8	9
1	3,14,15,21,36								
2	4,33,34	2,4,15,21,41							
3			2,5,16,31,41						
	2,15,35								
4				21,22,28,36					
			22,34,41						
5				8,12,28	5,7,13,33,47				
6	4,12					5,25,33,47			
					5,8,21				
7	4,12,37						5,19,25,47		
					5,21,25	5,37			
8				8,12,28				7,33,47	
9									33,47
								12,38,39	
10	4,12,19								
								12,38,39	8,17,19
11					3,8,19	8,19	8,12,19	8,21,38	12,18,20
			5,8,17						
12				8,12,28	3,8,19	8,12,19	8,12,19	8,21,38	12,18,20
			5,17,25						
13									
14									
15									
16									
17									
18									
19									
20									

21	4,37,38	4,12,37			12,19,24				
			4,5,25						
22				9,10,19				9,38,39	9,38,39
23				9,10,19				9,38,39	9,38,39
24				9,10,19				9,38,39	9,38,39
25									
26									
27									
28									
29									
30	2,4,6	2,5,8	2,6,7						
31									

References

[1] Brombacher AC. Quality and reliability problems from a customer's perspective: an increasing problem overlooked. *Quality and Reliability Engineering International* 2006*;* **22**: 821-838.

[2] Overton D. No Fault Found returns cost the mobile industry $4.5 billion per year. Media Bulletin, Investigating the mobile "No Fault Found Phenomenon", WDS Global 2006. http://www.wdsglobal.com; last accessed 08/04/2011.

[3] Koca A, Funk M, Karapanos E, Rozinat A, Martens J, Brombacher A. Soft Reliability: an Interdisciplinary Approach with a User-System Focus. *Quality and Reliability Engineering International* 2008; **25**(1): 3-20.

[4] Li TS, Huang HH. Applying TRIZ and fuzzy AHP to develop innovative design for automated manufacturing system. *Expert System with Application* 2009; **36**: 8302-8312.

[5] Cavallucci D, Khomenko N. From TRIZ to OTSM-TRIZ: Adressing complexity challenges in inventive design. *International Journal of Product Development* 2007; **4**: 4-21.

[6] Khomenko N, De Guio R, Lelait L, Kaikov I. A framework for OTSM-TRIZ based computer support to be used in complex problem management. *International Journal of Computer Applications in Technology* 2007; **30**: 88-104.

[7] Baldussu A, Becattini N, Cascini G. Network of contradictions analysis and structured identification of critical control parameters. *Procedia Emgineering* 2011; **9**: 3-17.

[8] Zanni-Merk C, Cavallucci D, Rousselot F. An ontological basis for computer aided innovation. *Computers in Industry* 2009; **60**: 563-574.

[9] Cong H, Tong LH. Grouping of TRIZ Inventive Principles to facilitate automatic patent classification. *Expert System with Applications* 2008; **38**: 788-795.

[10] Zhang J, Shang J. Research on Developing Environmental Protection Industry Based on TRIZ Theory. *Procedia Environmental Sciences* 2010; **2**: 1326-1334.

[11] http://www.triz40.com/ ; accessed 20/05/2011.

[12] Domb E. The 39 Features of Altshuller's Contradiction Matrix. http://www.triz-Journal.com/archives/1998/11/d/index.htm; accessed 20/05/2011.

[13] http://inventionmachine.com ; accessed 20/05/2011.
[14] http://www.trisolver.eu ; accessed 20/05/2011.
[15] http://ideationtriz.com/ ; accessed 20/05/2011.
[16] http://www.stat-design.com ; accessed 20/05/2011.
[17] Umar A, Tatari KK. Appropriate Web Usability Evaluation Method during Product Development. Master Thesis, Software Engineering, Thesis no: MSE-2008-03. School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden.
[18] Howarth J, Smith-Jackson T, Hartson R. Supporting novice usability practitioners with usability engineering tools. *Human computer studies* 2009; **67**: 533-549.
[19] Montabert C, McCrickard D, Winchester WW. An integrative approach to requirements analysis: how task models support requirements reuse in a user-centric design framework. *Interacting with computer* 2009; **21**: 304-315.
[20] Koca A, Funk M, Karapanos E, Rozinat A. Grasping product pragmatics: a case with internet on TV. *Proceeding of the 1st International Conference on Designing interactive user experiences for TV and video* 2008; 193-202. Silicon Valley, California, USA.
[21] Koca A, Brombacher A, Panchal J, Mistree F. Engineering soft reliability in product realization. *Proceeding of the ASME 2009*: International Design Engineering Technical Conferences 2009; 739-751.
[22] Wever R, Boks C, Van Kuijk JI. User-centred design for sustainable behavior. *International Journal of Sustainable Engineering* 2008; **1**: 9-20.
[23] Lilley D, Bhamra T, Lofthouse V. Towards sustainable use: an exploration of designing for behavioural change. *Proceeding of the DeSForM2006*: Design and semantics of form and movement 2006; 84-97.
[24] http://www.usabilitynet.org/ ; accessed 20/05/2011.
[25] Dix A, Finlay J, Abowd G, Beale R. *Human-Computer Interaction*. 2nd ed., Europe: Prentice Hall; 1998.
[26] Jin BS, Ji YG, Choi K, Cho G. Development of a usability evaluation framework with quality function deployment: from customer sensibility to product design. *Human Factor and Ergonomics in Manufacturing* 2009; **19**(2): 177-194.
[27] Verhaegen PA, Peeters J, Vandevenne J, Dewulf S, Duflou JR. Effectiveness of the PAnDA ideation tool. *Proceedings of the TRIZ future conference* 2010; 59-68.