INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

EXPRESS ENGINEERING CHANGE MANAGEMENT

Eli Kolberg[1], Yoram Reich[1] and Ilya Levin[2]
[1]School of Mechanical Engineering, Tel Aviv University
[2]School of Education, Tel Aviv University

ABSTRACT
Change is perhaps the most persistent aspect of product development. It could arise from external input that must be overcome but also from an explicit choice for improvement. In spite of this general appreciation, the management of engineering changes in development processes is lacking. Only recently this subject has started to attract a growing number of studies attempting to tame the impact of change on development processes. We propose that a plan for addressing change must include the use of carefully designed collection of design methods, termed design methodology. Such methodology must be tailored to the particular design context. We demonstrate through a case study, in the context of a high school mechatronics design course, that such a methodology leads to highly effective engineering change management. We conjecture that similar practices would be effective in industrial settings.

Keywords: design rationale, knowledge management, robotics, mechatronics, design methodology, agility, robust design, fault tolerant design

1 INTRODUCTION
Engineering changes are inherent to any engineering project. The need for change may arise due to many aspects. For example, fast technological change or inexperienced designers may lead to lack of technical knowledge. Alternatively, the need for quick time-to-market leads to employ concurrency in the design process, which in turn, instigates making design decisions without all necessary information. These situations will lead to engineering changes when new knowledge is gained about the product or new information arrives from parallel development paths, rendering previous decision defunct.
If changes arrive in the initial stages of the project the consequences might not be critical; however, some change requests at the end of a project could easily fail it or consume major effort and cost. Therefore, engineering change management is crucial to manufacturing organizations. From a state of minimal presence in research papers as reported by [10], the situation has changed in the last few years with publications dealing with industry practices [28],[29]; improving the administrative process of change management [16],[28]; and supporting the actual change handling [3],[4],[10],[25].
Fricke and Schulz [6] discussed the idea of designing for changeability. They propose that in order to support changes in systems throughout their life cycle, these systems' architecture needs to incorporate four aspects: Flexibility, agility, robustness, and adaptability. They further propose principles that support these aspects such as system simplicity and components independence. From our perspective, these principles should arise from the design methodology used in the design. A design methodology that needs to support changeability will use these and other principles as required by the context. Such methodology may include methods for improving changeability or resilience to change [24][27].
We propose that the first step in the ability to respond quickly to changes is an appropriate design methodology which is designed to address its context [22]. The context includes the technical aspect (e.g., what is the nature of the products being developed, which disciplines are involved in its design), the market aspect (e.g., who are the customers, is the market dynamic), the human aspect (e.g., who are the designers, how are teams organized), and the project resources (e.g., what is the design infrastructure, what is the project budget). When one of these aspects includes the need for addressing change, then if designed well, the design methodology has to support it.

Our study takes place in the context of a high school mechatronics course. The course lasts two years and central to it is the design, development, and testing in a competition of an autonomous mobile robot. The course has been running for several years. Its curriculum focused on the technical knowledge needed to design a robot. Students in the class have won prizes in an international robot competition; nevertheless, we observed that the products developed could be improved. Consequently, we set to improve the course by teaching them design methodology in addition to the technical knowledge. This methodology was designed to fit the context of the course and part of the requirements was to address the need for engineering change during the robot design. The drivers for addressing change included inexperienced designers, the short development time that required cycling through design, prototyping, and testing stages, as well as the desire to expose students to methods that support efficient change management.

We present a case study of handing a particular engineering change of a mobile robot just before it was scheduled to pass a qualifying test. The change was critical and had to be analyzed, solved, and implemented in hardware and software under extreme mental and objective time pressure. It was handled successfully in an astonishingly quick manner. This capability was afforded by the design methodology the designers mastered through the course. The case shows how the design methods comprising the methodology supported the change management. We argue that with minor modifications, similar design practices could support similar capabilities in industrial context.

2 THE CONTEXT OF THE CHANGE MANAGEMENT CASE

The context of the case study is a mechatronics course for high school students. In this course, students study mechatronics through a PBL approach [12],[22]. The course lasts 2 years in which high school students without technical background learn the technical subjects needed to build an autonomous robot. In addition to the technical subject, students are taught a 40-hour section on design methods. In the course, teams of students design and build autonomous mobile robots for an international competition. The competition is the fire fighting robot competition hosted annually by Trinity College, Hartford, CT (www.trincol.edu/events/robot). In the competition, a robot has to explore a model of a house, find a candle, extinguish it and return to the entry point. The competition rules are modified annually.

The case study provides an overview of the design of the robot **$ff-$01=$fe** that won the 1st place in the 2004 contest. The robot was designed and built by a team of 6 high school students. The case focuses on the change management exercised by the team on the qualification day before the competition when the team encountered the floor obstacle that was different from the contest rules. In a matter of two hours, the team analyzed the change; developed potential counter measures; analyzed their consequences; selected the solution strategy; and implemented and tested it successfully. No other team ventured to address the change.

2.1 The design methodology

Engineering design is the process of creating a solution to customer demand, whether observed or assumed, by an economic product. This process could be supported by a set of well orchestrated methods that form a coherent methodology. The methodology developed for the course context is composed of six methods [13]:

1. Problem formulation: ATR. Problem formulation is central to successful product development. The desire to fully understand the problem before starting design is seldom possible, the formulation and the designed product evolve simultaneously [2]. Atomic (which cannot be further divided to two or more requirements) Requirements (*ATR*s) is a tool that helps to understand the problem requirements, desired functionality and debugging requirements [11][26]; it allows to divide the requirements into very fundamental, thus simple to understand requirements. It also helps identify unnecessary, overlapping, or conflicting requirements, isolate bugs, and clarify what is to be done to implement the requirements. In the debugging mode, and problem solving, each requirement can be tested easily and separately. Using *ATR*s facilitate mutual understanding among programmers, testers, engineers, mentors, salesmen, management staff, and other persons who have to approve or to implement them, regardless of the *ATR*s' actual phrasing.

2. Conceptual design (CD), together with problem formulation is one of the two most critical steps in product development [31]. Tools for supporting *CD* are mainly intuitive. They range from idea

generation (e.g., brainstorming [19]), to a collection of structured methods (e.g., *QFD* [1]) for translating between requirements and engineering characteristics (e.g., the *HoQ*, house of quality), and subsequently, evaluating concepts (Pugh concept selection [20]) and questioning their performance (e.g., *FMEA* – Failure Mode and Effects Analysis). The latter is a proactive tool that enables the identification and prevention of process or product errors before they occur. New methods such as *SOS* also support concept generation [24],[31]. Using *CD* tools improves project organization, supports understanding the critical issues early on, and helps divide the work between participants. Consequently, *CD* systematically supports changeability.

3. ASIT (Advanced Systematic Inventive Thinking) is a systematic method for creative thinking, which is designed especially for problem solving [9]. It is important when a solution to a non-trivial problem is needed. The students see that by using this method, they can solve complicated problems and not just in the course but also in their life in general. *ASIT* encourages multidisciplinary thinking, generalization and integration capabilities and allows distinguishing between the essential and unimportant problem aspects.

4. Microprogramming (μP) allows for designing the robot control by handling an interface between the robot (operational unit) and the robot's controller (control unit). The *μP* method allows designing the control by using two different representations of the control part: Finite State Machine (FSM) and Algorithmic State Machine (ASM) that make it easy for designing, debugging, and coding, simultaneously [14]. *μP* design shows the students that there is a duality between two representations of control schemes (FSM and ASM) and that even though it is more "natural" to use one to describe the robot operation (FSM), it is better to use another in order to be more robust and efficient (ASM) [7][14]. This duality is particularly useful in dealing with change as the control could be modified in the FSM representation while making it more efficient in the ASM representation. *μP* also shows them a way for being more effective when for example it is possible to combine two or more control schemes and save resources.

5. Fault tolerance (FT) brings insight of the difference between products that are designed according to requirements, and robust products that can sustain faults up to a certain degree. It also introduces the students with possible faults during the design phase which improves their ability to identify and overcome problems. This influences the students to be more careful when they design the robot parts, for example, the sensor array. It also demonstrates that in unstructured environments, no design could survive without making it robust to faults because we are unable to foresee all potential situations.

Another issue addressed by *FT* is introducing the idea of a design verification starting from early design stages. This idea further translates into performing testing of finished product as well as on-line and off-line testing and the self-checking [7]. Additionally, *FT* includes a so-called design-for-testability (*DfT*). The main idea of the *DfT* is taking into account the *FT* issues in advance, and designing robots having an ability to be tested. Actually, providing the testability (comprising of observability and controllability) is the correct way to design any reliable technical system. To be fault tolerant, a system must be able to overcome permanent, intermittent or transient faults. The testing or self-checking capabilities need to be translated into alleviating or eliminating the impact of the fault. Adding the *FT* property into a system design may complicate it but if faults go undetected, autonomous robots in real-world environments may behave in an unpredictable or dangerous manner [18]. This creates a trade-off for product designers.

6. Fuzzy logic (*FL*) simplifies issues related to motors control. It is more straightforward and can be checked in an easy way, compared to other control methods. *FL* control is more intuitive to the students and is faster to implement than other control methods. Moreover, Gundogdu and Erenturk [8] show that the results obtained from the fuzzy controller of a DC motor driven four-bar mechanism are not only functionally superior than an optimal PID controller, but also much better in the controller output signal structure, which is more remarkable in terms of the hardware implementation. *FL* led to improved control reliability, under uncertainty conditions that is easily adaptable to new problems.

2.2 The design project

Following discussion with their mentors and graduates of the course the team decided to divide itself into hardware group composed of four students and software group with two students. There was continuous interaction between the groups; for example, if the software team designed the robot navigation in a way that required two distance sensors on each of the robot sides, it dictated constraints

on the robot structure. On the other hand, since the robot size is limited by the competition rules, the hardware team took into account the limitations enforced by the hardware and stability considerations when the sensors' location was designed. Figure 1 provides an overview of the design process.

Initially, the team studied the competition rules and extracted requirements that were gradually refined to the list in Table 1 **(**step **{1}** in Figure 1**)**. Each of the above subjects was further analyzed for obtaining more detailed "atomic" requirement. For example, in relation to the driving/steering system (1h), the students raised additional requirements (Table 2).

These requirements assisted the students to appreciate the complete task. For example, the students realized that a smaller robot could have more space to recover from undesired situations like hitting the wall. That directed them to design a small footprint robot (requirement 1h.11, Table 2) and to increase its height for the needed hardware space, while maintaining reasonable robot stability. Figure 3 shows some of these design parameters and their relationships.

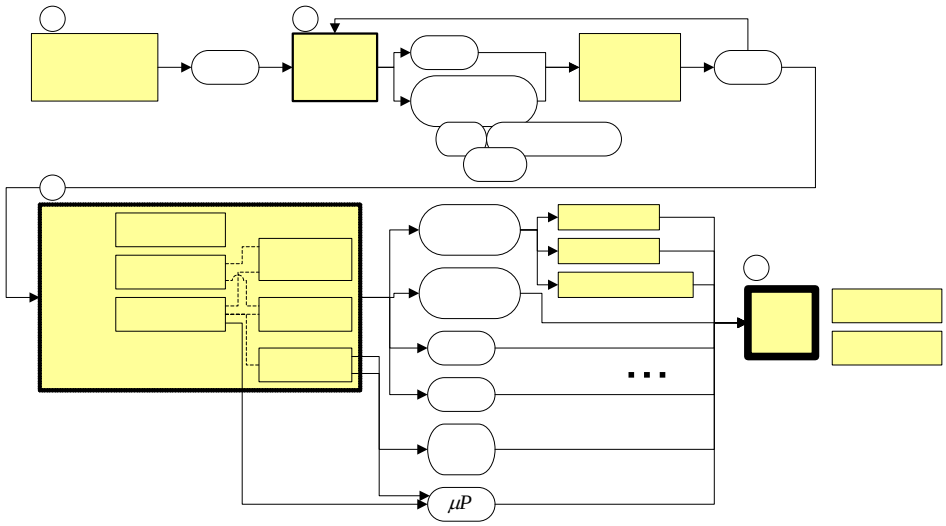


Figure 1: Overview of the design process.

Table 1: ATR final version (condensed)

1. Autonomous robot a) Microcontroller b) Motors & drivers c) Construction materials d) Sensors e) Battery f) Electricity system g) Extinguishing device h) Driving/steering design i) Sensors' array design j) Wiring design k) Maintenance design l) Control design m) Algorithms & software design **2. Wall following**	**3. No tethered operation** **4. Fastest possible** a).....Weight b).....Center of gravity c).....Motors power d).....Uneven floor consideration e).....Furniture consideration f) .....Stop conditions **5. Sound activation** **6. Furniture** **7. Uneven floor handling** a) Speed considerations b) Balance & center of gravity c) Wheelbase d) Possible locations	**8. Candle extinguishing** **9. Return trip** **10. No touch of wall** a) Smaller robot b) Special algorithms c) Escape possibilities d) Straight wall following e) Rotations f) Speed and position control **11. No touch of candle** **12. Reliability** a) Fault tolerance b) Easy and fast repairs c) Valid software d) Possible fast changes

Table 2: Refinement of driving/Steering ATR (1h)

1. Manoeuvrability when making turns 2. Ability for fast recovery from the corridor middle straight line 3. Ability to overcome inclined surfaces in a reliable way 4. Ability to avoid furniture with easy manoeuvre 5. As small as possible turn radius 6. As small as possible correction after doing a turn 7. Stable and will not crush if it hits a wall, furniture, or	8. Functional enable aligning of the robot at all room entrances 9. Simple and easy to implement system 10. Ability of the software team to deal properly with the chosen driving/steering system 11. Smallest possible footprint to allow more space to recover from a fault, or a wide turn before hitting the wall

inclined surface	

After the requirement refinement, the students started using conceptual design tools (step **{2}** in Figure 1; details in Figure 2**)**. They divided the requirements into four robot main subsystems (mechanics, electronics, software, and control)[1] and proceeded with each subsystem, considering its own requirements, using tools like *QFD*, *FMEA*, and failure analysis. For example, they carefully considered 11 possible driving/steering alternatives from which they selected a forward driving steering; this selection recorded all pros and cons of the alternatives and the reasons for the final choice. Subsequent discussions focused on decreasing overall robot footprint size (requirement 1h.11).

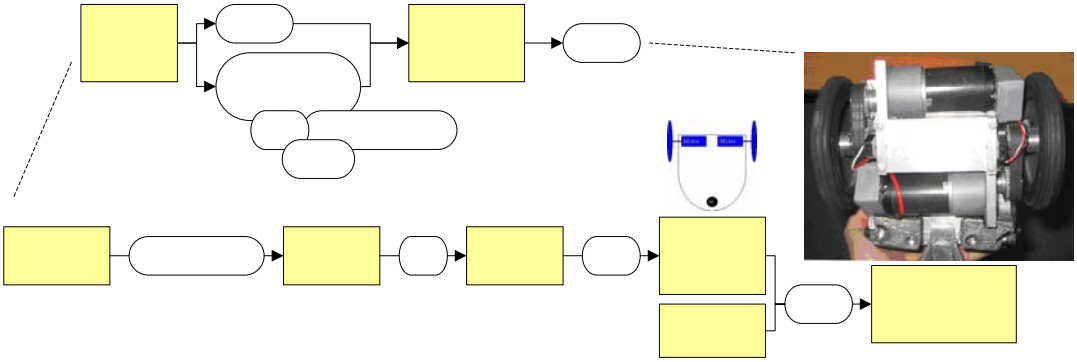
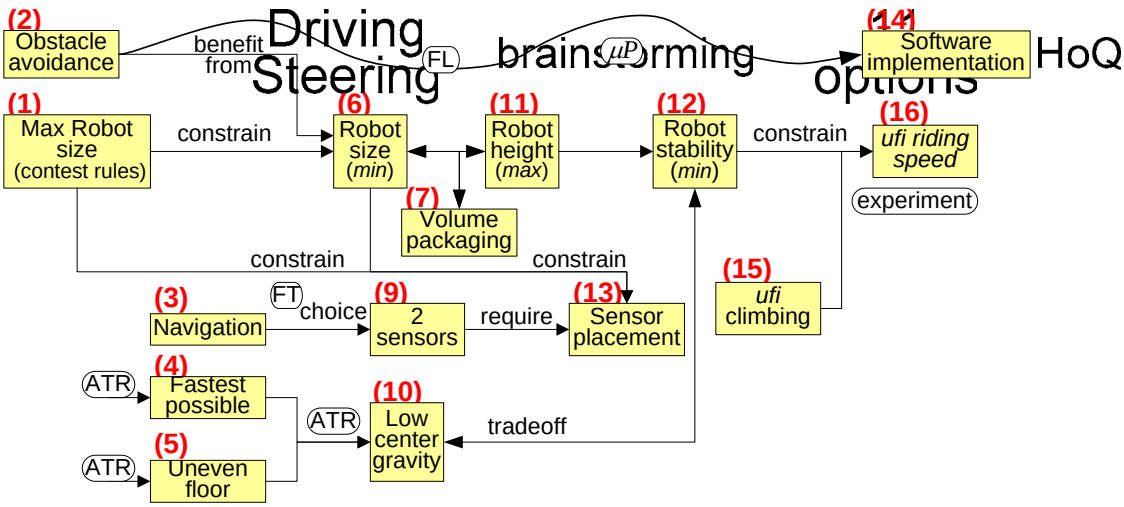


Figure 2: Details of conceptual design of drive mechanism

The motors were the most influential component on the robot footprint. Reducing motors size would allow for a significant decrease of robot size which would increase the free space for robot movement between the corridor walls, thereby improving robot reliability. For this task, the student used *ASIT*. They followed the process, selecting the solution strategy (restructuring) and the particular techniques of breaking symmetry and focussed on the motors. After a detailed session, they developed a compact asymmetric drive (Figure 2), with its advantages and disadvantages which they had to further consider. A critical issue in the design is the history keeping for facilitating future modifications. By systematically using *QFD* and other structured tools for almost every significant decision (e.g., selecting driving mechanism, extinguishing mechanism, and programming language, and placement of sensors), the students kept records of all options, their consequences, final choices, and subsequent modifications. In essence, such structured methods facilitate recording design rationale [21]. In future modifications, the sequence of tool use could be visited to check whether any decision needs to be revised. In addition, students kept detailed records of the relationships between different problem parameters while they performed the design; a model of such relationships is depicted in Figure 3.



[1] By the time the students finished with steps {2} and {3}, their initial subdivision into four main subsystems has changed to the one depicted in Figure 1, *ATR* detail ③.

Figure 3: Relationship between robot design parameters

Following the conceptual design, the students could continue refine the requirements **(**step **{3}** Figure 1**)**. For example, now that the drive has been selected, the requirement to 'not hit a wall' (refer to 10c in Table 1) could refer to the selected hardware as sown in Table 3.

Table 3: Refinement of ATR 10c

a. Stop robot	1. clear right motor speed, 2. send brake data to the motor controller for right motor, 3. clear left motor speed, 4. send brake data to the motor controller for left motor. 5. wait 100μs.
b. Drive robot backwards	6. turn on right motor. 7. send reverse direction to right motor. 8. send slow speed to right motor. 9. turn on left motor. 10. send reverse direction to left motor. 11. send slow speed to left motor. 12. send reverse direction to right motor. 13. wait 1 sec. 14. clear right motor speed, 15. send brake data to the motor controller for right motor, 16. clear left motor speed, 17. send brake data to the motor controller for left motor.

*ATR*s such as those in Table 3 could easily be translated into software control statements by using the *μP* conventions **{9}**. *μP* was used to design the software. Basically the robot system was divided into a control unit (CU) and operation unit (OU) along with a memory, which allowed for clear identification of input (*x*) to the CU from the OU, and output (*Y*) from the CU to the OU. This information enabled creating a finite state machine (FSM) of the robot control and functions that is easy to debug and to allow for easy modifications as necessary. The development of the software itself was easier to execute in the equivalent representation as an ASM [15],[17]. Therefore, short portions of the software, like right wall navigation, searching for the candle flame, etc., were first constructed as ASMs, and then converted to FSMs for further debugging and ease of handling.

The final ATR list (④ Figure 1) also included many important requirements such as making the robot more reliable through the use of the following tools:

1. Failure analysis **{5}** of past designs helped avoiding past failures such as wiring schemes problems, use of one battery to the fan, logic, driving/steering subsystems, choosing the fan controller, and the failures of past designs in navigation and turns in the arena.

2. FMEA **{4}** helped in identifying possible failure of the IR sensors due to two possible distances for each sensor's output voltage; fan placement in front; choice of extinguishing device; and choice of distance sensors.

3. Fault tolerance **{6}** was integrated with the hardware conceptual and detailed design. For example it helped in designing stable robot by lowering the robot's center of gravity, reduced the wires' length for obtaining a better organized layout of the robot, and lowered the height of the UV sensor in order to prevent false readings. In addition, subsequent to the software design, *FT* used to add checkers for sensors' readings, battery voltage, and detection of illegal micro-instructions.

4. Fuzzy logic **{8}** was used to design the robot speed and position control. The simplicity of the rules provided a way to concurrently design the hardware and control. For example, at first the students used one distance sensor in the robot front, because there was no need to align the robot front; only side alignments was required. Then, in the control design stage, one of the fuzzy control rules was:

> "if the motor is spinning much too slow and the motor speed is slowing down a lot and if the motor is keeping accumulating low speed a lot, then the motor speed should be increased a lot".

Then when using *FT*, there was a need to catch a situation where the robot front is too close to the wall. So the rule was changed to:

"if the motor is spinning much too slow and the motor speed is slowing down a lot and if the motor is keeping accumulating low speed a lot and if the robot is not too close to a front wall, then the motor speed should be increased a lot".

That forced the team to design front wall identification. As there was no way to know if the robot will be aligned with the front wall, or with left or right deviation, the team decided to add another sensor at the robot front so there would be a right and left front distance sensors that will identify close proximity of the robot to a front wall, no matter how the robot is approaching the wall.

The simplicity and readability of the control rules and the fact that they connected several design parameters, allowed using them for tracing the impact of changes in design parameters on others.

5. *ASIT* **{7}**. When the students encountered a difficult problem such as various tradeoffs, they employed *ASIT* to provide guidance. For example, *ASIT* was used for IR sensor geometrical placement, improving the IR sensors' unstable readings, placing the fan, and improving the flame focusing sensor function.

The whole design process could be described as a problem formulation exercise. Starting from the original contest rules (① in Figure 1), the requirements have been transformed into general *ATR*s (②), became better understood and more detailed *ATR*s (③), by proposing conceptual solutions that facilitated additional refinement. Subsequently, by fleshing out all outstanding issues, elaborating unmet requirements, and by making sure that all the requirements could be met by the proposed design, they became the final *ATR*s (④).

3 THE CHANGE MANAGEMENT CASE

3.1 The problem

On Saturday, the earliest time possible for qualification purpose, a requirement for participating in the contest, the team noticed that the uneven floor items (*ufi*) were different than their description published in the official contest web site. The robot, which was designed for different items, would not perform well with these modified items. Figure 4(a) shows the floor item building instructions as were introduced by the contest official web site. Figure 4(b) presents the actual *ufi* used in the contest. The organizer covered the *ufi* with black material in such a way that it made them higher, wider, and changed their original shape.

It was clear to the students that the robot would not function well with these new *ufi* since they remembered its input into the planned maneuvering of the robot. They had only two hours before their qualifying run and decided to change the robot. As shown in Figure 3, each requirement or design parameter influenced and was influence by multiple other parameters. Consequently, any change was expected to produce a ripple effect. Yet, the team was confident. The software team said it would not be a difficult problem, as they have built the software using *µP* and *FT* guidelines. Therefore, it will take no more than two hours including experiments to modify the software. This was in fact true, it took twenty minutes for the hardware team and then after one hour and thirty minutes, the robot was fully functional and handled perfectly the new *ufi*.

(a) (b)

Figure 4: Floor item: (a) official item, (b) shape with black covering (bottom view)

3.2 The change handling

In order to understand fully how to address this change, the team had to start from the change request to the original requirements as they developed from ① to ④ and see how the change affects other decisions. The students had to be very confident that they could handle the change including possible hardware and software modifications in the available timeframe. No other team attempted to do so.

It quickly became clear that the problem caused by the *ufi* change was reduced robot stability while climbing *ufi*. In the robot design, the students faced a similar problem related to robot stability when using *FMEA*. There, they reduced the robot footprint to facilitate better fault recovery and addressed the stability by arranging heavy parts below. The options they considered in the past were: increase robot size; add weight; lower robot height; or rearrange robot parts. These options were either infeasible (lower height); too difficult to implement (rearrange robot part would involve change of wiring including power lines extensions, changing sensor placements, etc., which could not be realized in two hours); or harm other performance qualities (add weight would require more power or would reduce robot speed). Given this deadlock, the student initiated an *ASIT* session that took 20 minutes and whose template is shown in Table 4. The conclusion was extending the castor wheel for improving the longitudinal stability while maintaining the transverse dimension as before. Operating within the 'closed world' principle of *ASIT*, guided the team to make the least change to the robot so that it could be implemented in two hours.

Table 4: ASIT session template

I. Preparation Stage	
	Problem objects list: Motors, wheels, connectors, and base plate.
	Neighbourhood objects list: Robot components, arena, and floor items
	Functional structure: The robot has to overcome the new floor items. The object that decreases the robot ability to handle the new floor items is robot size.
II. Solution Stage	
	Operation: Increasing robot size
	Restructuring technique selection: unification
	Conceptual solution: The relation “decrease the robot ability to handle the new floor items,” will change from decrease relation to increase if the following operation of “increasing the robot size” will be performed.
	Select an object: castor wheel
	Solution statement: The object “castor wheel” will be modified so that it will increase robot size.

The task was now to analyze the details of the choice and make sure that all its consequences could be handled effectively within the available timeframe and resources. The first task was determining the precise length of the wheel extension. This involved merely modifying previous calculation with the new *ufi* dimensions, leading to a 2*cm* extension of the longitudinal robot dimension. As a result, a search was conducted to find all items in the design that incorporated this or derivative dimensions as well as items that could be affected by the change of the *ufi* dimensions. This search was done in the context of *FMEA* to make sure that not only previously addressed issues are visited and resolved but any new impact of the change is handled. The search included top-down and bottom-up inspections.

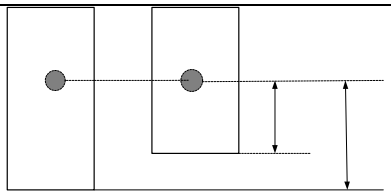
In the top-down inspection, the different versions of the robot requirements (i.e., ① - ④) were checked one by one. The in-room activity of finding the candle and extinguishing it were not affected as this dimension played no role in their consideration. The requirements that were affected were (see Table 1): Driving/steering design (1h) and Uneven floor handling - speed considerations (8a), balance and center of gravity (8b), escape possibilities (10c), rotations (10e), speed and position (10f), fault tolerance (12a), and valid software (12c). In relation to the steering (see Table 2), requirements (1h.1), (1h.3), (1h.4), (1h.5), (1h.6), (1h.7), (1h.10), and (1h.11) were affected.

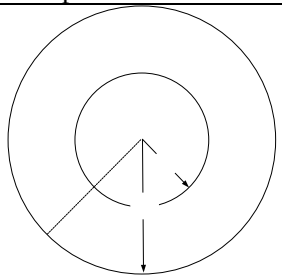
Now a focused bottom-up inspection was done to collect the detailed equations or decisions involving the longitudinal dimension. Among all the above requirements, the issues that incorporated this dimension, the robot activities that were found to be affected were the navigation and the *ufi* riding. The students divided the analysis into two parts accordingly.

3.2.1 Uneven floor item riding

In the initial robot design, while using *FMEA*, the students found that robot turns might cause faults (requirement 1h.1). The robot's dimensions and its center of rotation were the factors influencing on these faults. Subsequently, in step **{2}** of Figure 1 (detailed in Figure 2), the team decided upon small

robot dimensions. Changing the longitudinal dimension required revisiting previous calculations (Figure 5). The students calculated the distance that the castor wheel will travel for each 1º of rotation about the robot center and then calculated the difference between the present and previous dimensions.





$l_1 = 11cm$ - Initial radius of rotation of the robot

$l_2 = 13cm$ - Robot's radius of rotation after extending the castor wheel

$x_1 = \frac{2 \times \pi \times l_1}{360} = \frac{2 \times \pi \times 11}{360} = 0.2cm$ - The radial distance that the robot before extending the castor wheel move for each 1º of rotation

$x_2 = \frac{2 \times \pi \times l_2}{360} = \frac{2 \times \pi \times 13}{360} = 0.227cm$ - The radial distance that the robot after extending the castor wheel move for each 1º of rotation

Figure 5: The robot's turns and corrections

A correction to the robot direction would be performed when a deviation of 30º from the corridor centreline occurs. In that case, the distance difference would be $30 \times (0.227 - 0.2) = 0.81cm$. Unless addressed, this difference of less than $1cm$ might cause the robot to bounce.

In addition, the *ufi* contribution to the inclined angle of the robot had to be fixed (requirement 1h.3). The relative added correction due to the distance from the center of gravity was $(0.227 - 0.2)/0.2 = 0.135$. The added correction due to the *ufi* incline is 25% and is caused by the difference between the *ufi* height of the higher driving wheel (3cm) and the lower driving wheel (~2.5 cm). This difference causes the two driving wheels to travel different distances and this difference needs to be corrected to maintain straight line drive. The principle idea was to have the castor wheel on the ground (before hitting the *ufi*) while the front driving wheels are already located on or beyond the *ufi* top surface. Overall, the correction needed to be larger than the previous correction of the robot in the corridor with additional 25+13.5=38.5%. This correction was incorporated in the control rules. This and subsequent corrections of described herein where easily performed as their location in the software was easily identified because the software was developed using *µP* that followed *ATR*s.

On the *ufi* surface, there are sharp changes in the driving direction and more moderate direction changes perpendicular to the driving direction. With a speed of 0.45 m/s, the robot drives over the first inclined plane of the *ufi* in its wider side (about 2.5cm) in a time of $0.025m/0.45m/s = 0.055s$. When the robot climbs upward, within a short time it becomes unstable. This has been experienced in the previous design and recorded in a way similar to the one modelled by items (12), (15), and (16), in Figure 3. Therefore, it was clear that there was a need for slower speed and immediate corrections. Similar to the experiments they did in the previous design, the students performed experiments in order to find the proper speed of the robot and the difference between the wheels speed for correction purposes, when the robot is over the *ufi*. The speed found to be $0.2m/s$. This value changed the previous value in the control rules.

The students also had to figure out how the robot is going to behave when encountering an *ufi*. Adding new hardware like low height distance sensor, a gyroscope, a compass, tilt sensor, etc., was out of the question because it would cause too much change and add further needed adaptations. In addition, the students wanted to keep the "closed world" principle of *ASIT*. The solution the students came out with was using the fact that when the robot first goes over an *ufi* it makes fast turn in an angle much larger than the usual deviation when the robot drives on a flat surface. So all it takes is to add to the $10ms$ real time interrupt routine check of the robot angle, which is basically the difference between the side sensors. When this difference is greater then a threshold number, the robot will go to an *ufi* mode with reduced speed and sharper correction. As soon as the difference will be less then the threshold number it will go to a normal operation. The common difference between the side sensors for correction in

normal operation was (2V-1.6V)=0.4V. The common difference between the side sensors for correction over the *ufi* was (2.4V-1.2V)=1.2V. These calculations were familiar to the students from the design of the robot. The threshold value was chosen to be 0.8V. This value was incorporated in the control rules.

3.2.2 Navigation

It was clear that the change in the longitudinal dimension did not change navigation along a straight wall because the sensors location remained intact. The only issue that had to be addressed was turning (requirements 1h.1, 1h.3, 1h.4, 1h.5, 1h.9). The significant change after increasing the longitudinal dimension was related to turns. In order to successfully execute the turn, the distance of the robot from the wall in front and the turn radius should be modified as presented in Figure 6.

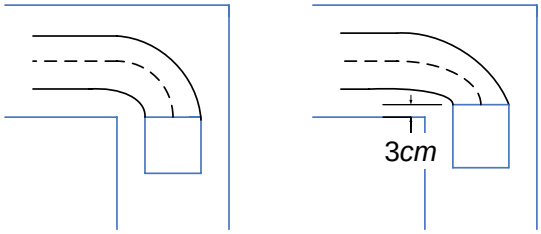


Figure 6: Different turns of the robot before (left) and after (right) castor wheel extension

In the previous design, the students used *FT* to analyze the turning of the robot around corners. If the robot starts the turn late, its front would be close to the wall in front and while turning, its front wheel or castor wheel could hit this wall. If however, it starts too early, it might hit the opposite wall. The students made several calculations and conducted experiments for finding the range of distance to the wall in front that led to successful turn. They did so for several robot dimensions in order to evaluate which dimension to prefer.

After the change, the students only needed to check the new dimension against their previous calculations, test the robot in turning and pick the range that was successful in the experiments. Following these experiments, they picked the threshold value for distance from front wall to be 32*cm* (49) instead of 35*cm* (40). Consequently, the robot would start the turn when its front side has already entered 3*cm* into the intersection (Figure 6). After finding the new threshold, the particular control software was easy to modify in the right place.

4 DISCUSSION

Change is a fundamental property of product development processes. In a competitive world, the ability to manage change effectively is highly desirable. This capability needs to be set as an organizational goal and its attainment must be planned for carefully. In this paper we presented a case in which the need to address change was introduced and a set of design methods were collected to support mechatronics design as well as change management capability. This set of methods complement each other, overlap in ways that allows their smooth integration into a single methodology. The methods are structured and simple. They provide support starting from a problem statement up to the detailed design of hardware and software. For every decision or sub problem, these methods helped organize information, explore the space, and prepare measures to address faults or changes.

Maintaining a temporal record of methods utilization (e.g., as shown in Figure 1 and Figure 2) including all the information used, the choices made, and their reasons, allows for replaying previous decisions. A record of the influences between parameters (e.g., as shown in Figure 3) allows to identify decisions that may be influenced by design changes. Keeping organized calculations (e.g., Figure 5) allows quick recalculation of parameters. In addition, the use of methods such as *FMEA* and *FT*, allows additional inspection of the potential consequences of changes.

The implementation of changes is also made easy by the use of *ATR*s. They represent basic design information that could easily be located and modified. By keeping the traceability between different refinement levels of *ATR*s, it is also easy to locate potentially influenced requirements that were missed by using the previous process.

Two of the methods are specifically built to support evolutionary development. *μP* allows moving between two representations (ASM and FSM) in which one (FSM) is easier for conceptualization and debugging and the other is easier for implementation (ASM). Similarly, *FL* provides support for evolutionary development of control strategy. Altogether, the six design methods provide diverse support for change management.

The capability to manage change afforded by the methodology was demonstrated in a stressful situation that required fast response. Out of all the competing teams that faced the need to address the change, only the team analyzed in here ventured to manage it and did it well and with confidence.

We argue that the same tools, tailored to different design contexts would provide similar change management capabilities. However, this generalization needs to be demonstrated in different design situations. One way to empirically test this generalization would be in educational settings in which change requests would be planned and students' responses would be analyzed.

The applicability of the methods to industrial practice needs to be qualified as well. The design task described in this case study was for the team, the development of a new product. As such, the team generated most of the relevant knowledge for developing the product (except, for example, past designs that were analyzed by the team). In industry, many design projects are evolutionary development of previous products. Such projects have their own requirements and constraints and the methodology presented would create a trace of decisions from the initial requirements to the final product. In such cases, the methodology will assist in making changes to new design decisions. As time passes, additional decisions would be recorded in new product versions or through reverse engineering of old products, thus gradually increasing the applicability of the methodology to address changes.

This study is directly related to design rationale capture techniques and the motivation that underlie the development of design rationale capture methods. Similarly, the process described before for managing this process is essentially knowledge management. Consequently, we could have presented the methodology as a design rationale capture or knowledge management practices.

REFERENCES

[1] Akao Y. (ed), *Quality Function Deployment*, Productivity Press, Cambridge, MA, 1990.

[2] Braha D. and Reich Y., Topological structures for modeling engineering design processes, *Research in Engineering Design*, 14(4):185-199, 2003.

[3] Clarkson P.J. and Hamilton J.R., 'Signposting', a parameter-driven task-based model of the design process, *Research in Engineering Design*, 12(1):18-38, 2000.

[4] Eckert C., Clarkson P.J., Zanker W., Change and customisation in complex engineering domains, *Research in Engineering Design*, 15(1):1-21, 2004.

[5] Fricke E., Gebhard B., Negele H., Igenbergs E., Coping with changes: Causes, findings, and strategies, *Systems Engineering***,** 3(4):169-179, 2000.

[6] Fricke E. and Schulz A.P., Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle, *Systems Engineering*, 8(4):342-259, 2005.

[7] Graf B., Hans M., and Schraft R.D. Mobile robot assistants, *IEEE Robotics & Automation Magazine*, 11(2):67-77, 2004.

[8] Gundogdu O., and Erenturk K., Fuzzy Control of a DC Motor Driven Four-Bar Mechanism, *Mechatronics*, 15(4):423-438, 2005.

[9] Horowitz R. Creative Problem Solving in engineering Design, *PhD Thesis*, Tel-Aviv University, Faculty of Engineering, 1999.

[10] Huang G.Q. and Mak K.L., Current practices of engineering change management in UK manufacturing industries, *International Journal of Operations & Production Management*, 19(1):21-37, 1999.

[11] IEEE Guide for Developing System Requirements Specification, in *IEEE Standards, Software Engineering, Volume One, Customer and Terminology Standards*, IEEE-Std-1233, Computer Society, 1998a.

[12] Kolberg E., Reich Y., and Levin I., Project-based high school mechatronics course, *International Journal of Engineering Education*, 19(4):557–562, 2003.

[13] Kolberg E., Reich Y., and Levin I., Transforming design education by design, in *Proceedings of the 17th International Conference on Design Theory and Methodology (DTM)*, (New York, NY), ASME, 2005.

[14] Levin I., Kolberg E., and Reich Y., Robot control teaching with state machine based design method, *International Journal of Engineering Education*, 20(2):234–243, 2004.
[15] Levin I., Mioduser D. (1996). A Multiple-Constructs Framework for Teaching Control Concepts, *IEEE Transactions of Education*, 39(4):488-496.
[16] Loch C. H. and Terwiesch C. (1999) Accelerating the Process of Engineering Change Orders: Capacity and Congestion Effects. Journal of Product Innovation Management, 16(2):145-159.
[17] Mioduser D., Levin I. (1996). Cognitive-conceptual Model for Integration Robotics and Control into the Curriculum, *Computer Science Education*, 7(2):199-210.
[18] Murphy, R.R. Trial by Fire [Rescue Robot], *IEEE Robotics & Automation Magazine*, 11(3):50-61, 2004.
[19] Osborn A. F., *Applied Imagination*, Charles Scribner's Sons, New York, NY, 1953.
[20] Pugh S. *Total Design – Integrated Methods for Successful Product Engineering*, Addison-Wesley, GB, 1990.
[21] Reich Y., Improving the rationale capture capability of QFD, *Engineering with Computers*, 16:(3-4):236-252, 2000.
[22] Reich Y., Kolberg E., and Levin I., Designing contexts for learning design**,** *International Journal of Engineering Education*, 22(3):489-495, 2006.
[23] Reich Y. and Levy E., Managing product design quality under resource constraints, *International Journal of Production Research*, 42(13):2555–2572, 2004.
[24] Reich Y. and Ziv-Av A., Robust product concept generation, in *CDROM Proceedings of the 15th International Conference on Engineering Design (ICED)*, The Design Society, 2005.
[25] Rouibah K. and Caskey K.R., Change management in concurrent engineering from a parameter perspective, *Computers in Industry*, 50(1),15-34, 2003.
[26] Salzer, H., and Levin I. Atomic Requirements in Teaching Logic Control Implementation, *International Journal of Engineering Education*, 20, 1: 46-51, 2004.
[27] Sered Y. and Reich Y., Standardization and modularization driven by minimizing overall process effort, *Computer-Aided Design*, 38(5):405-416, 2006.
[28] Subrahmanian E., Lee C., Thomas M., and the *n*-dim group, Managing and supporting product life-cycle through engineering change management, *International Journal of Product Life Cycle Management, accepted for publication, 2006*.
[29] Terwiesch C., Loch C.H., Managing the process of engineering change orders: The case of the climate control system in automobile development, *Journal of Product Innovation Management*, 16(2):160-172, 1999.
[30] Wright I.C., A review of research into engineering change management: implications for product design, *Design Studies*, 18:33-4, 1997.
[31] Ziv-Av A. and Reich Y., SOS – Subjective objective system for generating optimal product concepts, *Design Studies*, 26(5):509-533, 2005.

Contact: Y. Reich
Tel Aviv University
School of Mechanical Engineering
Tel Aviv 69978
Israel
Phone: +972-3-6407385
Fax: +972-3-6407617
e-mail: yoram@eng.tau.ac.il
URL: http://www.eng.tau.ac.il/~yoram