Concurrent Engineering of Robot Manipulators

M. Reza Emami and Robin Chhabra
University of Toronto Institute for Aerospace Studies
Canada

1. Introduction

Robot manipulators are good examples of complex engineering systems, where designers occasionally employ a subsystem-partitioning approach for their analysis and synthesis. The design methodology is traditionally based on the sequential decomposition of mechanical, electromechanical, and control/instrumentation subsystems, so that at each step a subset of design variables is considered separately (Castano et al., 2002). Although conventional *decoupled* or *loosely-coupled* approaches of design seem intuitively practical, they undermine the interconnection between various subsystems that may indeed play a crucial role in multidisciplinary systems. The necessity of communication and collaboration between the subsystems implies that such systems ought to be synthesized concurrently. In the concurrent design process, design knowledge is accumulated from all the participating disciplines, and they are offered equal opportunities to contribute to each state of design in parallel. The *synergy* resulting from integrating different disciplines in concurrent design has been documented in several case studies, to the effect that the outcome is a new and previously unattainable set of performance characteristics (Hewit, 1996). However, the challenge in a concurrent design process is that the multidisciplinary system model can become prohibitively complicated; hence computationally demanding. Plus, a large number of multidisciplinary objective and constraint functions must be taken into account, simultaneously, with a great number of design variables. As the complexity of the system model increases, in terms of the interactions between various subsystems, the coordination of all the constraints distributed in different disciplines becomes more difficult, in order to maintain the consistency between performance specifications and design variables.
Within the context of robotics, several *ad hoc* techniques of concurrent engineering have been reported in the literature. They are innovative design schemes for specific systems, such as Metamorphic Robotic System (Chirikjian, 1994), Molecule (Rus & McGray, 1998), Miniaturised Self-Reconfigurable System (Yoshida et al., 1999), Crystalline (Rus & Vona, 2000), and Semi-Cylindrical Reconfigurable Robot (Murata et al., 2000). But, more systematic approaches have been suggested by other researchers beyond the robotics community to tackle the challenge of high dimensionality in concurrent design. These approaches can be divided into two major groups. The first group translates the model complexity into a large volume of computations, and then attempts to find efficient algorithms or parallel

processing techniques to make these computations feasible. For example, parallel genetic algorithms were used for multi-objective optimizations (Coello, 1999), and later augmented with a penalty method to handle constraints (Kurapati et al., 2000). This approach was later adopted for the concurrent engineering of modular robotic systems (Bi & Zhang, 2001). Also, an integration of agent-based methods and simulated annealing was used for the modular configuration design (Ramachandran & Chen, 2000). The second group tries to alleviate the complexity by reducing the optimization space; either through breaking the optimization process into several stages (Paredis, 1996), or by approximating the space with the one with lower dimensions (Dhingra & Rao, 1995). Each group brings certain contributions to concurrent engineering, yet cannot avoid some drawbacks. While efficient algorithms, mostly taking advantage of parallel processing, can handle high computational demands in concurrent engineering, they tend to lose transparency, so that designers can no longer relate to the process. On the other hand, a better understanding of design may be achieved, should one be able to simplify the optimization model, but at a great cost of obtaining outcomes for an approximated version of the system that can be far from reality.

This chapter introduces a solution for the complexity of concurrent engineering, which in essence consists of two unique constituents, each relating to one of the above-mentioned groups. For the first part, it utilizes an efficient system modeling technique that not only does not compromise the transparency, but also accounts for complex phenomena such as sensor noise, actuator limitation, transmission flexibility, etc., which can hardly be captured by computational modeling. The model efficiency, in terms of both computation and accuracy, is due to the use of real hardware modules in the simulation loop and, hence, the real-time execution. In other words, the solution uses a Robotics Hardware-in-the-loop Simulation (RHILS) platform for "computing" the system model in the design process. And for the second part, the solution applies an alternative design methodology, namely Linguistic Mechatronics (LM), which not only formalizes subjective notions and brings the linguistic aspects of communication into the design process, but also transforms the multi-objective constrained optimization model into a single-objective unconstrained formulation. A combination of the above two techniques will ensure an efficient solution for concurrent engineering of robot manipulators, without simplifying the system model. Further, it facilitates communication between designers (of different background) and customers by including linguistic notions in the design process.

The chapter is organized as follows: Section 2 introduces Linguistic Mechatronics (LM). Section 3 details the Robotics Hardware-in-the-loop Simulation (RHILS) platform. Section 4 describes the LM-RHILS based concurrent engineering methodology and its application to an industrial robot manipulator. Some concluding remarks are made in Section 5.

2. Linguistic Mechatronics: An Alternative Approach to Concurrent Engineering

The premise of concurrent engineering is to provide a common language to fill in the communication gap between different engineering disciplines, and to devise a means for helping them collaborate towards a common goal. The need for communication and collaboration in concurrent engineering implies that, in addition to physical features, many subjective notions must be involved, which can hardly be captured by pure mathematical formulations. Both customers and designers need to communicate beyond the equations to

convey design requirements and specifications. Hence, there is a need for a communication means in concurrent engineering that can convey qualitative and subjective notions that are used frequently in human interactions, in addition to holistic criteria that finalize the design process based on objective performances in the real physical world. A few methodologies of concurrent design have attempted to include subjective notions in the design process (e.x., Dhingra et al., 1990). Amongst them, Method of Imprecision (MoI) is a notable attempt to take into account imprecision in design (Otto & Antonsson, 1995). This approach defines a set of designer's preferences for design variables and performance parameters to model the imprecision in design. It determines and maximizes the global performance under one of the two conservative or aggressive design tradeoff strategies, and uses fuzzy-logic operators for tradeoff in the design space. This method offers a number of advantages that are crucial in concurrent engineering. However, it does not provide a systemic means to distinguish the constraints from the goals in the aggregation process; instead it simply offers two extreme designer's attitudes. Further, in the MoI methodology designer's attitudes are not justified with any objective performance criterion. While subjective notions can play a crucial role in concurrent engineering of multidisciplinary systems, their relevance must eventually be checked against the objective criteria of system performance.
This section introduces *Linguistic Mechatronics* (LM) as an alternative concurrent design framework, which emphasizes on the designer's satisfaction, instead of pure performance optimization, and brings the linguistic aspects of communication into the design process. It not only formalizes subjective notions of design and simplifies the complicated multi-objective constrained optimization, but also resolves the above-mentioned deficiencies of the MoI methodology through **a)** dividing the design attributes into two inherently-different classes, namely *wish* and *must* attributes; and **b)** aggregating satisfactions using *parametric* fuzzy-logic operators so that the designer's attitude can be adjusted based on an objective performance criterion. Linguistic Mechatronics involves three stages of system modeling. First, a fuzzy-logic model is developed in the *primary* phase of design; secondly, a software and/or hardware simulation of the system is used for the *secondary* phase. And lastly, a bond graph model of the system assigns appropriate *supercriteria* that finalize the design. In the following sub-sections, the foundations of linguistic mechatronics, namely fuzzy modeling and fuzzy operators, will be reviewed first, and then a step-by-step formulation of the LM methodology will be presented.

2.1 Fuzzy-Logic Modeling

Fuzzy-logic modeling is an approach to forming a system model by using a descriptive language based on fuzzy-logic with fuzzy propositions. In (Emami, 1997), a systematic approach of fuzzy-logic modeling is developed, which is adopted in this work. In general, the clustered knowledge of a system can be interpreted by fuzzy models consisting of IF-THEN rules with multi-antecedent and multi-consequent variables (n antecedents, s consequents, and r rules):

$$\begin{array}{ll}
\textbf{IF}\ U_1 \text{ is } B_{11}\ \textbf{AND}\ldots\textbf{AND}\ U_n \text{ is } B_{1n}\ \textbf{THEN}\ V_1 \text{ is } D_{11}\ \textbf{AND}\ldots\textbf{AND}\ V_s \text{ is } D_{1s} & \\
\textbf{ALSO} & \\
\qquad\qquad \ldots & (1) \\
\textbf{ALSO} & \\
\textbf{IF}\ U_1 \text{ is } B_{r1}\ \textbf{AND}\ldots\textbf{AND}\ U_n \text{ is } B_{rn}\ \textbf{THEN}\ V_1 \text{ is } D_{r1}\ \textbf{AND}\ldots\textbf{AND}\ V_s \text{ is } D_{rs} &
\end{array}$$

where U_j (j=1,…,n) is the j^{th} input variable and V_k (k=1,…,s) is the k^{th} output variable, B_{ij} (i=1,...,r, j=1,…,n) and D_{ik} (i=1,…,r, k=1,…,s) are fuzzy sets over the input and output universes of discourse, respectively. Constructing a fuzzy model can be divided into two major steps: **a)** fuzzy rule-base generation, and **b)** fuzzy inference mechanism selection.

A. Fuzzy Rule-base Generation

Assuming the existence of sufficient knowledge of the system, the process of rule-base generation can be performed in the following sequence: **a)** clustering output data and assigning output membership functions, **b)** finding the non-significant input variables and assigning the membership functions to the rest of them, and **c)** tuning the input and output membership functions. Clustering methods are occasionally based on the optimization of an objective function to find the optimum membership matrix, $\boldsymbol{U}=[u_{ik}]$, that contains the membership value of the k^{th} data point, $\boldsymbol{z}_k \in \boldsymbol{Z}$, to the i^{th} partition. In Fuzzy C-Means (*FCM*) clustering method, this function, J_m, is defined as the weighted sum of the squared errors of data points, and the minimization problem is formulated as:

$$\min_{(\mathbf{U},\mathbf{V})}\left[J_m(\boldsymbol{U},\boldsymbol{V};\boldsymbol{Z}) = \sum_{i=1}^{c}\sum_{k=1}^{N}(u_{ik})^m(\boldsymbol{z}_k-\boldsymbol{v}_i)^T(\boldsymbol{z}_k-\boldsymbol{v}_i)\right]; \tag{2}$$

where $\boldsymbol{V}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,\dots,\boldsymbol{v}_c\}$ is the set of unknown cluster centers, N and c are the number of data points and clusters, respectively, and m is the weighting exponent.

A prerequisite for *FCM* is assigning c and m. The optimal values of these numbers are calculated based on two requirements: **a)** maximum separation between the clusters; and **b)** maximum compactness of the clusters. Therefore, the *fuzzy within-cluster scatter matrix*,

$$\boldsymbol{S}_W = \sum_{i=1}^{c}\sum_{k=1}^{N}(u_{ik})^m(\boldsymbol{z}_k-\boldsymbol{v}_i)(\boldsymbol{z}_k-\boldsymbol{v}_i)^T \tag{3}$$

and *between-cluster scatter matrix*,

$$\boldsymbol{S}_B = \sum_{i=1}^{c}\left(\sum_{k=1}^{N}(u_{ik})^m\right)(\boldsymbol{v}_i-\bar{\boldsymbol{v}})(\boldsymbol{v}_i-\bar{\boldsymbol{v}})^T \tag{4}$$

are defined to reflect the two criteria (Emami et al., 1998). Note that the fuzzy total mean array, $\bar{\boldsymbol{v}}$, is defined as:

$$\bar{\boldsymbol{v}} = \frac{1}{\sum_{i=1}^{c}\sum_{k=1}^{N}(u_{ik})^m}\sum_{i=1}^{c}\sum_{k=1}^{N}(u_{ik})^m\boldsymbol{z}_k . \tag{5}$$

The matrix $\boldsymbol{S}_B$ represents the separation between the fuzzy clusters, and $\boldsymbol{S}_W$ is an index for the compactness of fuzzy clusters. For obtaining the best clusters the trace of matrix $\boldsymbol{S}_W$, $tr(\boldsymbol{S}_W)$, should be minimized to increase the compactness of clusters and $tr(\boldsymbol{S}_B)$ should be maximized to increase the separation between clusters. Alternatively, $s_{cs} = tr(\boldsymbol{S_W}) - tr(\boldsymbol{S_B})$ can be minimized to identify the optimum number of clusters, *c*. The weighting exponent, *m*, varies in $(1,+\infty)$ and indicates the degree of fuzziness of the assigned membership functions. In order to have a reliable index for s_{cs}, *m* should be far enough from both extremes. Hence, the reliable value of *m* is what holds the trace of *fuzzy total scatter matrix* ($\boldsymbol{S}_T$),

$$s_T = tr(\boldsymbol{S_T}) = tr(\boldsymbol{S_W} + \boldsymbol{S_B}), \tag{6}$$

somewhere in the middle of its domain. Since s_T and s_{cs} are both functions of *m* and *c*, the process of choosing the parameters should be performed by a few iterations.
In systems with a large number of variables, there occasionally exist input variables that have less effect on the output, in the range of interest. In order to have an efficient fuzzy-logic model, an index, π_j, is defined as an overall measure of the non-significance of input variable x_j as:

$$\pi_j = \prod_{i=1}^{n} \frac{\Gamma_{ij}}{\Gamma_j}; \qquad (j=1,\ldots,r) \tag{7}$$

where Γ_{ij} is the range in which membership function $\mathrm{B}_{ij}(x_j)$ is one, and Γ_j is the entire range of the variable x_j. The smaller the value of π_j is, the more effect the jth variable has in the model, and vice versa.
Finally, to map the output membership functions onto the input spaces, a clustering method, called *line fuzzy clustering*, is employed. This method works based on the distance of each data point located on the axis x_j, to the interval of the jth input variable corresponding to the output membership function equal or close to one (Emami, 1997).

B. Fuzzy Reasoning Mechanism
To interpret connectives in fuzzy set theory, there exist a number of different classes of triangular norm (*t-norm*) and triangular conorm (*t-conorm*), such as *Max-Min Operators* (T_{min}, S_{max}), *Algebraic Product and Sum* ($T_{prod}\ S_{sum}$), and *Drastic Product and Sum* (T_W, S_W). Using the basic properties of these operators, it is shown in (Emami, 1997) that for any arbitrary *t-norm* (*T*) and *t-conorm* (*S*) and for all $a_i \in [0,1]$:

$$\begin{aligned} T_W(a_1,\ldots,a_n) &\le T(a_1,\ldots,a_n) \le T_{\min}(a_1,\ldots,a_n), \\ S_W(a_1,\ldots,a_n) &\le S(a_1,\ldots,a_n) \le S_{\max}(a_1,\ldots,a_n). \end{aligned} \tag{8}$$

Various types of parameterized operators have been suggested in the literature to cover this range. In particular, a class of operators for fuzzy reasoning is introduced in (Emami et al., 1999), which is adopted here for aggregating the satisfactions, as explained in the next sub-section:

$$S^{(p)}(b_1, b_2, \ldots, b_n) = [b_1{}^p + (1-b_1{}^p)[\ldots[b_{n-2}{}^p + \ldots + (1-b_{n-2}{}^p)[b_{n-1}{}^p + (1-b_{n-1}{}^p)b_n{}^p]]\ldots]]\ ^{1/p}; \tag{9}$$

where $b_i \in [0,1]$ and $p \in (0,+\infty)$. Consequently, the corresponding *t-norm* operator is defined based on De Morgan laws using standard complementation operator, as:

$$T^{(p)}(a_1, a_2, ..., a_n) = 1 - S^{(p)}((1-a_1),(1-a_2),...,(1-a_n)) . \tag{10}$$

In the extreme cases, this class of parameterized operators approaches (T_{min},S_{max}) as $p \to +\infty$, (T_{prod} S_{sum}) as $p \to 1$, and (T_W, S_W) as $p \to 0$.

The meaning of an aggregation operator is sometimes neither pure AND (*t-norm*) with its complete lack of compensation, nor pure OR (*t-conorm*). This type of operator is called *mean aggregation* operator. For example, a suitable parametric operator of this class, namely *generalized mean operator*, is defined in (Yager & Filev, 1994) as:

$$G^{(\alpha)}(a_1, a_2, ..., a_n) = \left(\frac{1}{n} \sum_{i=1}^{n} a_i^{\alpha} \right)^{1/\alpha} ; \tag{11}$$

where $\alpha \in (-\infty,+\infty)$. It appears that this type of aggregation monotonically varies between *Min* operator while $\alpha \to -\infty$ and *Max* operator as $\alpha \to +\infty$. Subsequently, an appropriate inference mechanism should be employed to combine the rules and calculate the output for any set of input variables. Takagi-Sugeno-Kang (*TSK*) reasoning method is associated to a rule-base with functional type consequents instead of the fuzzy sets and the crisp output, y^* , is defined by the weighted average of the outputs of individual rules, y_i's, as:

$$y^* = \sum_{i=1}^{r} \frac{\tau_i}{\sum_{j=1}^{r} \tau_j} y_i = \sum_{i=1}^{r} \frac{\tau_i}{\sum_{j=1}^{r} \tau_j} (b_{i0} + b_{i1}x_1 + ... + b_{in}x_n) ; \tag{12}$$

where τ_i is the degree of fire of the ith rule:

$$\tau_{\mathrm{i}} = T(\mathrm{B}_{i1}(x_1),...,\mathrm{B}_{in}(x_n)) . \tag{13}$$

Since the *TSK* method of reasoning is compact and works with crisp values, it is computationally efficient; and therefore, it is widely used in fuzzy-logic modeling of engineering systems, especially when tuning techniques are utilized. Ultimately, the parameters of input membership functions and output coefficients are tuned by minimizing the mean square error of the output of the fuzzy-logic model with respect to the existing data points.

2.2 The LM Formulation

A design problem consists of two sets: *design variables* $\boldsymbol{X} \equiv \{X_j : \forall j = 1,...,n\}$ and *design attributes* $\boldsymbol{A} \equiv \{A_i : \forall i = 1,...,N\}$. Design variables are to be configured to satisfy the *design* *requirements* assigned for design attributes, subject to the design availability $\boldsymbol{D} \equiv \{D_j : \forall j = 1,...,n\}$. Each design attribute stands for a design function providing a functional mapping $F_i : \aleph \to \Im_i$ that relates a state of design configuration $\boldsymbol{X} \in \aleph$ to the attribute $A_i \in \Im_i$, i.e., $A_i = F_i(\boldsymbol{X})$ (i=1,…,N). These functional mappings can be of any form, such as closed-form equations, heuristic rules, or set of experimental or simulated data.

Given a set of design variables and a set of design attributes along with an available knowledge that conveys the relationship between them, the process of Linguistic Mechatronics is performed in two phases: **a)** *primary* phase in which proper intervals for the design variables are identified subject to design availability, and **b)** *secondary* phase in which design variables are specified in their intervals in order to maximize an overall design satisfaction based on the design requirements and designer's preferences. Thus, the secondary phase involves a single-objective optimization, yet it is critically dependant on the initial values of a large number of design variables. The primary phase makes the optimization more efficient by providing proper intervals for the design variables from where the initial values are selected. The *overall satisfaction* is an aggregation of satisfactions for all design attributes. The satisfaction level depends on the designer's attitude that is modeled by fuzzy aggregation parameters. However, different designers may not have a consensus of opinion on *satisfaction*. Therefore, the system performance must be checked over a holistic *supercriterion* to capture the objective aspects of design considerations in terms of physical performance. Designer's attitude is adjusted through iterations over both primary and secondary phases to achieve the enhanced system performance. Therefore, this methodology incorporates features of both human subjectivity (i.e., designer's intent) and physical objectivity (i.e., performance characteristics) in multidisciplinary system engineering.

Definition 1 - Satisfaction: A mapping μ such that $\mu : \boldsymbol{Y} \to [0,1]$ for each member of $\boldsymbol{Y}$ is called satisfaction, where $\boldsymbol{Y}$ is a set of available design variables or design attributes based on the design requirements. The grade one corresponds to the ideal case or the most satisfactory situation. On the other hand, the grade zero means the worst case or the least satisfactory design variable or attribute.

Satisfaction on a design attribute, $a_i \equiv \mu_{A_i}(\boldsymbol{X})$, indicates the achievement level of the corresponding design requirement based on the designer's preferences. The satisfaction for a design variable, $x_j \equiv \mu_{X_j}(\boldsymbol{X})$, reflects the availability of the design variable. In the conceptual phase, design requirements are usually subjective concepts that imply the costumer's needs. These requirements are naturally divided into *demands* and *desires*. A designer would use engineering specifications to relate design requirements to a proper set of design attributes. Therefore, in *LM* the design attributes are divided into two subsets, labeled *must* and *wish* design attributes.

Definition 2 - *Must* design attribute: A design attribute is called *must* if it refers to costumer's demand, i.e., the achievement of its associated design requirement is mandatory with no room for compromise. These attributes form a set coined $\boldsymbol{M}$.

Definition 3 - *Wish* design attribute: A design attribute is called *wish* if it refers to costumer's desire, i.e., its associated design requirement permits room for compromise and it should be achieved as much as possible. These attributes form a set coined $\boldsymbol{W}$.
Therefore,

$$\boldsymbol{M} \cap \boldsymbol{W} = \phi, \qquad \boldsymbol{M} \cup \boldsymbol{W} = \boldsymbol{A}. \tag{14}$$

The satisfaction specified for *wish* attribute W_i is $w_i(\boldsymbol{X}) \equiv \mu_{W_i}(\boldsymbol{X})$ (i=1,…,N_W), and the satisfaction specified for *must* attribute M_i is $m_i(\boldsymbol{X}) \equiv \mu_{M_i}(\boldsymbol{X})$ (i=1,…,N_M). Therefore, for each design attribute A_i (corresponding to either M_i or W_i), there is a predefined mapping to the satisfaction a_i (m_i or w_i), i.e., $\{(A_i, a_i) : \forall i = 1,\ldots,N\}$. Fuzzy set theory can be applied for defining satisfactions through fuzzy membership functions and also for aggregating the satisfactions using fuzzy-logic operators.
Remark: $[F_i(\boldsymbol{X}_1) \succeq F_i(\boldsymbol{X}_2)] \Leftrightarrow [a_i(\boldsymbol{X}_1) \geq a_i(\boldsymbol{X}_2)]$ for monotonically non-decreasing satisfaction. More specifically, if $0 < a_i(\bullet) < 1$ then $[F_i(\boldsymbol{X}_1) \succ F_i(\boldsymbol{X}_2)] \Leftrightarrow [a_i(\boldsymbol{X}_1) > a_i(\boldsymbol{X}_2)]$ and if $a_i(\bullet) = 0 \text{ or } 1$ then $[F_i(\boldsymbol{X}_1) \succ F_i(\boldsymbol{X}_2)] \Leftrightarrow [a_i(\boldsymbol{X}_1) = a_i(\boldsymbol{X}_2)]$, where $\succeq$ denotes loosely superior and $\succ$ represents strictly superior. In other words, the better the performance characteristic is the higher the satisfaction will be, up to a certain threshold.
Definition 4 - Overall satisfaction: For a specific set of design variables $\boldsymbol{X}$, overall satisfaction is the aggregation of all *wish* and *must* satisfactions, as a global measure of design achievement.

A. Calculation of Overall Satisfaction

Must and *wish* design attributes have inherently-different characteristics. Hence, appropriate aggregation strategies must be applied for aggregating the satisfactions of each subset.

1) Aggregation of Must Design Attributes

Axiom 1: Given *must* design attributes, $\{(M_i, m_i) : \forall i = 1,\ldots,N_M\}$, and considering component availability, $\{(D_j, x_j) : \forall j = 1,\ldots,n\}$, the overall *must* satisfaction is the aggregation of all *must* satisfactions using a class of *t-norm* operators.

Must attributes correspond to those design requirements that are to be satisfied with no room of negotiation, and, linguistically, it means that all design requirements associated with *must* attributes have to be fulfilled simultaneously. Therefore, for aggregating the satisfactions of *must* attributes an AND logical connective is suitable. Considering satisfactions as fuzzy membership degrees, the AND connective can be interpreted through a family of *t-norm* operators. Thus, the overall *must* satisfaction is quantified using the *p*-parameterized class of *t-norm* operators, i.e.,

$$\mu_{\boldsymbol{M}}^{(p)}(\boldsymbol{X}) = T^{(p)}(m_1, m_2, \ldots, m_{N_M}, x_1, x_2, \ldots, x_n). \quad (p > 0) \tag{15}$$

The parametric *t-norm* operator $T^{(p)}$ is defined based on (9) and (10).

Parameter *p* can be adjusted to control the fashion of aggregation. Changing the value of *p* makes it possible to obtain different tradeoff strategies. The larger the *p*, the more pessimistic (conservative) designer's attitude to a design will be, and vice versa.

2) Aggregation of Wish Design Attributes

Definition 5 - Cooperative *wish* attributes: A subset of *wish* design attributes is called cooperative if the satisfactions corresponding to the attributes all vary in the same direction when the design variables are changed.

Therefore, *wish* attributes can be divided into two cooperative subsets:

a) Positive-differential *wish* attributes ($\boldsymbol{W}^+$): In this subset the total differential of the satisfactions for the *wish* attributes (with respect to design variables) are non-negative.

$$\boldsymbol{W}^+ = \{(W_i, w_i) : W_i \in \boldsymbol{W}, \quad dw_i(\boldsymbol{X}) \geq 0\}. \tag{16}$$

This subset includes all attributes that tend to reach a higher satisfaction when all design variables have an infinitesimal increment.

b) Negative-differential *wish* attributes ($\boldsymbol{W}^-$): In this subset the total differential of the satisfactions for the *wish* attributes (with respect to design variables) are negative.

$$\boldsymbol{W}^- = \{(W_i, w_i) : W_i \in \boldsymbol{W}, \quad dw_i(\boldsymbol{X}) < 0\}. \tag{17}$$

This subset includes all attributes that tend to reach a lower satisfaction when all design variables have an infinitesimal increment.

$$\boldsymbol{W}^+ \cap \boldsymbol{W}^- = \phi, \quad \boldsymbol{W}^+ \cup \boldsymbol{W}^- = \boldsymbol{W}. \tag{18}$$

Since in each subset all *wish* attributes are cooperative, their corresponding design requirements can all be fulfilled simultaneously in a linguistic sense. Hence, according to **Axiom 1**, similar to *must* satisfactions, a *q*-parameterized class of *t-norm* operators is suitable for aggregating satisfactions in either subsets of *wish* attributes.

$$\mu_{W^\pm}{}^{(q)}(\boldsymbol{X}) = T^{(q)}(w_1, w_2, ..., w_{N_{W^\pm}}) \quad (q > 0); \tag{19}$$

where $N_{W^\pm}$ are the number of positive-/negative-differential *wish* attributes.

Axiom 2: Given the satisfactions corresponding to positive- and negative-differential *wish* attributes, $\mu_{W^+}{}^{(q)}(\boldsymbol{X})$ and $\mu_{W^-}{}^{(q)}(\boldsymbol{X})$, the overall *wish* satisfaction can be calculated using an *a*-parameterized *generalized mean* operator.

The two subsets of *wish* attributes cannot be satisfied simultaneously as their design requirements compete with each other. Therefore, some compromise is necessary for aggregating their satisfactions, and the class of *generalized mean* operators in (11) reflects the averaging and compensatory nature of their aggregation.

$$\mu_W{}^{(\alpha,q)}(\boldsymbol{X}) = \left[\frac{1}{2}\left(\left(\mu_{W^+}{}^{(q)}(\boldsymbol{X})\right)^{\alpha} + \left(\mu_{W^-}{}^{(q)}(\boldsymbol{X})\right)^{\alpha}\right)\right]^{1/\alpha}. \tag{20}$$

This class of *generalized mean* operators is monotonically increasing with respect to *a* between *Min* and *Max* operators; therefore, offers a variety of aggregation strategies from conservative to aggressive, respectively. The overall *wish* satisfaction is governed by two parameters *q* and *a,* representing subjective tradeoff strategies. They can be adjusted appropriately to control the fashion of aggregation. The larger the *a* or the smaller the *q*, the more optimistic (aggressive) one's attitude to a design will be, and vice versa.

3) Aggregation of Overall Wish and Must Satisfactions

Axiom 3: The overall satisfaction is quantified by aggregating the overall *must* and *wish* satisfactions, $\mu_M{}^{(p)}(\boldsymbol{X})$, and $\mu_W{}^{(q,\alpha)}(\boldsymbol{X})$, with the *p*-parameterized class of *t-norm* operators, i.e.,

$$\mu^{(p,q,\alpha)}(\boldsymbol{X}) = T^{(p)}(\mu_M{}^{(p)}(\boldsymbol{X}), \mu_W{}^{(q,\alpha)}(\boldsymbol{X})). \quad (p > 0). \tag{21}$$

The aggregation of all *wish* satisfactions can be considered as one *must* attribute, i.e., it has to be fulfilled to some extent with other *must* attributes with no compromise. Otherwise, the overall *wish* satisfaction can become zero and it means none of the *wish* attributes is satisfied, which is unacceptable in design. Therefore, the same aggregation parameter, *p,* that was used for *must* attributes should be used for aggregating the overall *wish* and *must* satisfactions. In (21), three parameters, i.e., *p, q* and *a,* called *attitude parameters*, govern the overall satisfaction.

B. Primary Phase of LM

Once the overall satisfaction is calculated, in order to obtain the most satisfactory design, this index should be maximized. The optimization schemes are critically dependent on the initial values and their search spaces. Therefore, to enhance the optimization performance, suitable ranges of design variables are first found in the primary phase of *LM*. In linguistic term, primary phase of *LM* methodology provides an *imprecise* sketch of the final product and illustrates the decision-making environment by defining some ranges of possible solutions. For this purpose, the mechatronic system is represented by a fuzzy-logic model based on (1). This model consists of a set of fuzzy IF-THEN rules that relates the ranges of design variables as fuzzy sets to the overall satisfaction; i.e.,

IF X_1 is B_{11} **AND**…**AND** X_n is B_{1n} **THEN** μ is D_1
ALSO
… (22)
ALSO
IF X_1 is B_{r1} **AND**…**AND** X_n is B_{rn} **THEN** μ is D_r

where μ is the overall satisfaction and B_{lj} and D_l (j=1,…,n and l=1,…,r) are fuzzy sets on X_j

and μ, respectively, which can be associated with linguistic labels.
The fuzzy rule-base is generated from the available data obtained from simulations, experimental prototypes, existing designs or etc., using fuzzy-logic modeling algorithm as detailed in the previous section. The achieved consequent fuzzy sets, D$_l$'s, can be further defuzzified by (23) to crisply express the level of overall satisfaction corresponding to each rule.

$$\mu^*_l = \frac{1}{N}\sum_{i=1}^{N}\mu^i_l = \frac{1}{N}\sum_{i=1}^{N}(b_{l0} + b_{l1}X_1^{\ i} + \dots + b_{ln}X_n^{\ i}); \tag{23}$$

where μ^i_l (*l=1,2,…,r, i=1,2,…,N*) is the overall satisfaction corresponding to the *i*th data point in *l*th rule, *N* is the number of data points in the existing database, b_{lj} (*j=1,2,…,n*) is the *TSK* consequent coefficient corresponding to the *j*th design variable in the *l*th rule, $X_j^{\ i}$ is the *j*th design variable in the *i*th data point and μ^*_l corresponds to the overall satisfaction of rule *l*. The rule with the maximum μ_l^* is selected, and the set of its antecedents represents the appropriate intervals for the design variables. The set of these *suitable* intervals is denoted as $\boldsymbol{C} = \{C_j : \forall j = 1,\dots,n\}$ and the corresponding fuzzy membership functions are labeled as $c_j(X_j)\,(j = 1,\dots,n)$. Finally, these fuzzy sets are defuzzified using *Centre of Area* (*CoA*) defuzzification method (Yager & Filev, 1994) to introduce the set of initial values $\boldsymbol{X_0} = \{X_{j0} : \forall j = 1,\dots,n\}$ for design variables in the secondary phase of optimization process.

$$X_{j0} = \frac{\int_{C_j} X_j c_j(X_j)dX_j}{\int_{C_j} c_j(X_j)dX_j}. \quad (j = 1,\dots,n) \tag{24}$$

C. Secondary Phase of LM
In the secondary phase, *LM* employs regular optimization methods to perform a single-objective unconstrained maximization of the overall satisfaction. The point-by-point search is done within the suitable intervals of design variables obtained from the primary phase. Therefore, the locally unique solution $\boldsymbol{X_s}$ is obtained through:

$$\mu^{(p,q,\alpha)}(\boldsymbol{X}_s) = \max_{\boldsymbol{X}\in\boldsymbol{C}} T^{(p)}(\mu_M^{\ (p)}(\boldsymbol{X}), \mu_W^{\ (q,\alpha)}(\boldsymbol{X})). \tag{25}$$

It can be shown that the pareto-optimality of the solution is a result of how the satisfactions are defined: Assume that $\boldsymbol{X_s}$ is not locally pareto-optimal. Then $\exists \boldsymbol{X_1} \in \boldsymbol{C}$ such that

$$F_i(\boldsymbol{X_1}) \succeq F_i(\boldsymbol{X}_s), \quad \forall i = 1,\dots,N \tag{26}$$

particularly, there exists an i_0 that:

$$F_{i_0}(\boldsymbol{X_1}) \succ F_{i_0}(\boldsymbol{X}_s). \tag{27}$$

Thus, according to the **Remark**,

$$a_{i_0}(\boldsymbol{X_1}) > a_{i_0}(\boldsymbol{X_s}), \tag{28a}$$

or

$$a_{i_0}(\boldsymbol{X_1}) = a_{i_0}(\boldsymbol{X_s}) = 1. \tag{28b}$$

Hence, if F_{i_0} corresponds to a *must* attribute, due to the monotonicity of *t-norm* operator in (15),

$$\mu_{\boldsymbol{M}}{}^{(p)}(\boldsymbol{X_1}) \geq \mu_{\boldsymbol{M}}{}^{(p)}(\boldsymbol{X_s})\,. \tag{29}$$

And if F_{i_0} corresponds to a *wish* attribute, due to the monotonicity of both *t-norm* and *generalized mean* operators in (20),

$$\mu_{\boldsymbol{W}}{}^{(q,\alpha)}(\boldsymbol{X_1}) \geq \mu_{\boldsymbol{W}}{}^{(q,\alpha)}(\boldsymbol{X_s})\,. \tag{30}$$

Finally, the monotonicity of *t-norm* in (21) lead to:

$$\mu^{(p,q,\alpha)}(\boldsymbol{X_1}) \geq \mu^{(p,q,\alpha)}(\boldsymbol{X_s})\,. \tag{31}$$

Obviously, (31) contradicts the fact that $\boldsymbol{X_s}$ is a locally optimal solution. Note that in (29), (30) and (31) the equality holds when both satisfactions are 1. Thus, in order to avoid the equality, the satisfactions can be defined monotonically increasing or decreasing on the set of suitable intervals, $\boldsymbol{C}$.
As indicated in (25), various attitude parameters, *p, q* and *a*, result in different optimum design values for maximizing the overall satisfaction. Consequently, a set of satisfactory design alternatives ($\boldsymbol{C_s}$) is generated based on subjective considerations, including designer's attitude and preferences for design attributes.

D. Performance Supercriterion
From the set of optimally satisfactory solutions, $\boldsymbol{C_s}$, the best design needs to be selected based on a proper criterion. In the previous design stages, decision making was critically biased by the designer's preferences (satisfaction membership functions) and attitude (aggregation parameters). Therefore, the outcomes must be checked against a supercriterion that is defined based on physical system performance. Indeed, such a supercriterion is used to adjust the designer's attitude based on the reality of system performance. A suitable supercriterion for multidisciplinary systems should take into account interconnections between all subsystems and consider the system holistically, as the synergistic approach of mechatronics necessitates.

Although mechatronic systems are multidisciplinary, the universal concept of energy and energy exchange is common to all of their subsystems. Therefore, an energy-based model can deem all subsystems together with their interconnections, and introduce generic notions that are proper for mechatronics. A successful attempt in this direction is the conception of *bond graphs* in the early 60's (Paynter, 1961). Bond graphs are domain-independent graphical descriptions of dynamic behaviour of physical systems. In this modeling strategy all components are recognized by the energy they supply or absorb, store or dissipate, and reversibly or irreversibly transform. In (Breedveld, 2004; Borutzky, 2006) bond graphs are utilized to model mechatronic systems. This generic modeling approach provides an efficient means to define holistic supercriteria for mechatronics based on the first and second laws of thermodynamics (Chhabra & Emami, 2009).

1) Energy Criterion
Any mechatronic system is designed to perform a certain amount of work on its environment while the input energy is supplied to it. Based on the first law of thermodynamics, this *supplied energy* (*S*) does not completely convert into the *effective work* (*E*) since portions of this energy are either stored or dissipated in the system by the system elements or alter the global state of the system in the environment. This *cost energy* (*f*) should be paid in any mechatronic system in order to transfer and/or convert the energy from the suppliers to the effective work. Therefore, a supercriterion, coined *energy criterion*, can be defined as minimizing *f*(***X***) for a known total requested effective work from the system. Based on the principle of conservation of energy:

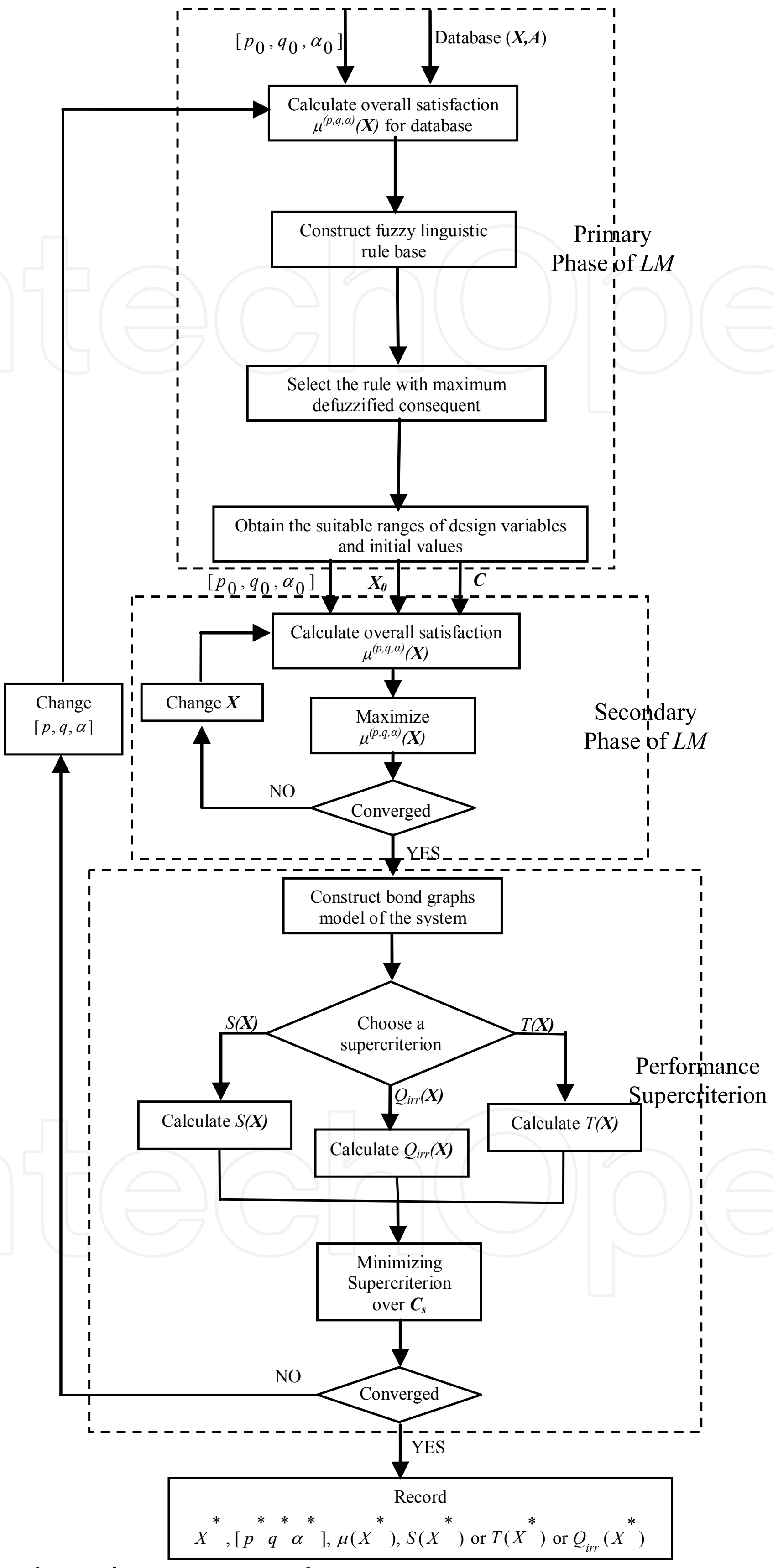
$$S(\boldsymbol{X}) = E + f(\boldsymbol{X}), \tag{32}$$

which shows that minimizing the supplied energy is equivalent to the energy criterion. Therefore, by minimizing the supplied energy or cost function, depending on the application, with respect to the attitude parameters the best design can be achieved in the set of optimally-satisfied solutions ($\boldsymbol{C}_s$).

$$S(\boldsymbol{X}^*) = \min_{\boldsymbol{X}_s \in \boldsymbol{C}_s} S(\boldsymbol{X}_s; p, q, \alpha). \tag{33}$$

In bond graphs the supplied energy is the energy that is added to the system at the source elements, which are distinguishable by S_e and S_f with the bonds coming out of them. Hence, by integrating the supplied power at all of the source elements during the simulation *S*(***X***) can be calculated.

2) Entropy Criterion
Based on the second law of thermodynamics, after a change in supplied energy, a mechatronic system reaches its equilibrium state once entropy generation approaches its maximum. During this period the system loses its potential of performing effective work, constantly. Therefore, if the loss work of the system is less, available work from the system or, in other words, the aptitude of the system to perform effective work on the environment is more. This is equivalent to minimizing the entropy generation or the irreversible heat exchange at the dissipative elements of the bond graphs, i.e., $Q_{irr}(t; \boldsymbol{X})$, with respect to ***X*** and accordingly it is called *entropy criterion*. Given a unit step change of supplied energy, the equilibrium time, denoted by $t_{eq}(\boldsymbol{X})$, is the time instant after which the rate of change of dissipative heat remains below a small threshold, *ε*,



$[p_0, q_0, \alpha_0]$
Database (X,A)
Calculate overall satisfaction $\mu^{(p,q,\alpha)}(X)$ for database
Construct fuzzy linguistic rule base
Primary Phase of LM
Select the rule with maximum defuzzified consequent
Obtain the suitable ranges of design variables and initial values
$[p_0, q_0, \alpha_0]$
X_0
C
Calculate overall satisfaction $\mu^{(p,q,\alpha)}(X)$
Change X
Change $[p, q, \alpha]$
Maximize $\mu^{(p,q,\alpha)}(X)$
Secondary Phase of LM
NO
Converged
YES
Construct bond graphs model of the system
Choose a supercriterion
S(X)
T(X)
$Q_{irr}(X)$
Performance Supercriterion
Calculate S(X)
Calculate $Q_{irr}(X)$
Calculate T(X)
Minimizing Supercriterion over C_s
NO
Converged
YES
Record
X^*, $[p^* \; q^* \; \alpha^*]$, $\mu(X^*)$, $S(X^*)$ or $T(X^*)$ or $Q_{irr}(X^*)$

Fig. 1. The flow chart of Linguistic Mechatronics

$$t_{eq}(\boldsymbol{X}) = Inf\{t_0 : \forall t > t_0 \; \dot{Q}_{irr}(t, \boldsymbol{X}) < \varepsilon\} . \tag{34}$$

Consequently, the best design is attained in the set of optimally satisfactory solutions,

$$Q_{irr}(t_{eq}(\boldsymbol{X}^*)) = \min_{X_s \in C_s} Q_{irr}(t_{eq}(\boldsymbol{X}_s); p, q, \alpha) . \tag{35}$$

3) Agility Criterion
Alternatively, for systems where response time is a crucial factor the rate of energy transmission through the system, or *agility*, can be used for defining the performance supercriterion. Thus, the supercriterion would be to minimize the time that the system needs to reach a steady state as the result of a unit step change of all input parameters at time zero. A system reaches the steady state when the rate of its *internal dynamic energy*, *K*, becomes zero. Internal dynamic energy is equivalent to the kinetic energy of masses in mechanical systems or the energy stored in inductors in electrical systems. Masses and inductors resist the change of velocity and current, respectively. In terms of bond graph modeling, both velocity and current are considered as *flow*. Consequently, internal dynamic energy is defined as the energy stored in the elements of system that inherently resist the change of *flow*. Therefore, Given a unit step change of input variables, the response time, denoted by $T(\boldsymbol{X})$, is the time instant after which the rate of change of internal dynamic energy, $\dot{K}$, remains below a small threshold, δ.

$$T(\boldsymbol{X}) = Inf\{t_0 : \forall t > t_0 \; \dot{K}(t, \boldsymbol{X}) < \delta\} . \tag{36}$$

As a design supercriterion, when the response time reaches its minimum value with respect to attitude parameters the best design is attained in $\boldsymbol{C_s}$.

$$T(\boldsymbol{X}^*) = \min_{X_s \in C_s} T(\boldsymbol{X}_s; p, q, \alpha) . \tag{37}$$

The complete flowchart of *LM* is presented in Fig. 1.

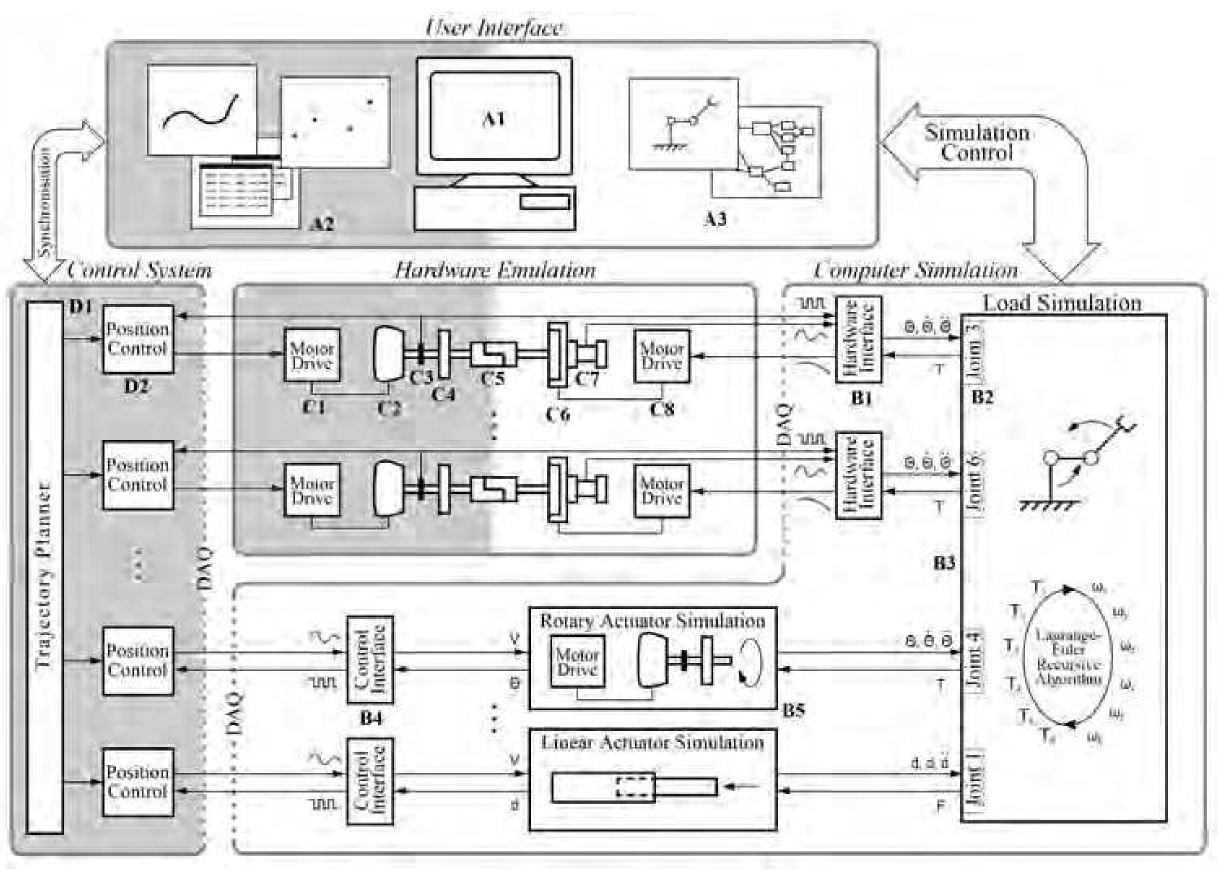
3. Robotic Hardware-in-the-loop Simulation Platform

The increasing importance of several factors has led to an increase in the use of HIL simulation as a tool for system design, testing, and training. These factors are listed in (Maclay, 1997) as: reducing development time, exhaustive testing requirements for safety critical applications, unacceptably high cost of failure, and reduced costs of the hardware necessary to run the simulation. By using physical hardware as part of a computer simulation, it is possible to reduce the complexity of the simulation and incorporate factors that would otherwise be difficult or impossible to model. Therefore, HIL simulations can play an effective role in systems concurrent engineering. The HIL simulations have been successfully applied in many areas, including aerospace (Leitner, 1996), automotive (Hanselman, 1996), controls (Linjama et al., 2000), manufacturing (Stoeppler et al., 2005), and naval and defense (Ballard et al., 2002). They have proven as a useful design tool that

reduces development time and costs (Stoeppler et al.; 2005; Hu, 2005). With the ever improving performance of today's computers it is possible to build HIL simulation without specialized and costly hardware (Stoeppler et al., 2005).
In the field of robotics, HIL simulation is receiving growing interest from researchers, and has been applied from a number of different perspectives. These approaches include: *robot-in-the-loop* simulations, such as the platform used for the task verification of the special-purpose dexterous manipulator at the Canadian Space Agency (Piedboeuf et al., 1999) or the use of both real and simulated mobile robots interacting with a virtual environment (Hu, 2005); *controller-in-the-loop* simulations, where a real control system interacts with a computer model of the robot (Cyril et al., 2000); and *joint-in-the-loop* simulations, which use a computer model to compute the dynamic loads seen at each joint and then emulate those loads on the real actuators (Temeltas et al., 2002). Each of these approaches applies the HIL concept slightly differently, but all have produced positive results. In a recent work (Martin & Emami, 2008), a modular and generic Robotic HIL Simulation (RHILS) platform was designed and developed for the industrial manipulators, and its performance was verified using the *CRS-CataLyst-5* manipulator from Thermo Fisher Scientific Inc. (Thermo, 2007). The RHILS platform was used in this work as the second constituent of robotic concurrent engineering, next to Linguistic Mechatronics. The architecture of the RHILS platform is illustrated in Fig. 2, and an overview of its modules is presented below:

3.1 RHILS Architecture

The RHILS platform architecture allows for simultaneous design and testing of both the joint hardware and control system of a robot manipulator. The architecture is designed to be adequately generic so that it can be applied to any serial-link robot manipulator system, and focuses on modularity and extensibility in order to facilitate concurrent engineering of a wide range of manipulators. This section presents a detailed breakdown of the main blocks of the architecture.
The architecture is separated into four subsystems: (a) the *User Interface*, (b) the *Computer Simulation*, (c) *Hardware Emulation*, and (d) the *Control System*, which are described below with reference to Fig. 2. These subsystems are further partitioned into two major categories: RHILS Platform components (indicated with a white background), and Test System components (indicated with a grey background). The RHILS Platform components are generic and should remain largely consistent over multiple applications, while the Test System components are part of the system being designed and/or tested on the platform. Depending on how much of the system is implemented in hardware versus how much is simulated it is possible to tailor the setup to all phases of the design cycle, and the architecture is designed to make adjusting this ratio as easy as possible.



A1 User interface host computer
A2 Control system user interface and trajectory setup
A3 Simulation user interface and scheduler
B1 Motor interface block, converts between actual hardware signals and the standardized form used in the simulation
B2 Joint assignment for the module
B3 Inverse dynamics simulation
B4 Control interface block, converts between actual control signals and the standardized form used with simulated actuators
B5 Simulated model of an actuator, for cases where the hardware module is unavailable, impractical, or unnecessary
C1 Drive electronics for Test Motor
C2 Test Motor
C3 Differential rotary encoder
C4 Harmonic drive transmission
C5 Detachable coupling to allow test hardware to be swapped in and out
C6 Load Motor
C7 Reaction torque transducer, for closed loop control and data acquisition
C8 Drive electronics for Load Motor
D1 Trajectory planner
D2 Position controller

A gray background indicates that section is part of the system being designed and tested using the RHIL platform

Fig. 2. RHILS Platform Architecture

A. User Interface Block

This block contains the most overlap between the RHILS Platform and the Test System. Because it is necessary to synchronize initial conditions before starting a simulation, this block acts as an intermediary between the custom control system and the generic simulation. On the RHILS Platform side robot configurations and parameters are chosen, as well as specifying any external conditions, for example zero-gravity or end-effector payloads, that will be used during a simulation. For the Test System side any configurable control parameters are set in the control system, such as the planned trajectories and feedback loop gains. Finally, the duration of the simulation and the type of data logging to be performed are selected.

B. Computer Simulation Block

The *Computer Simulation* performs three primary roles. Its first and most obvious task, represented by the *Load Simulation* block, is to run the inverse dynamics computations based on the instantaneous position, velocity, and acceleration of each joint, and solve for the dynamic load applied to each joint actuator. Due to the recursive algorithm used for computing the inverse dynamics (Li & Sankar, 1992) on the dedicated kernel, it is possible to specify any reasonable number of joints in any configuration and still attain the computational efficiency necessary to run the simulation in real-time. The second task is to convert the hardware signals read in and sent out through a data acquisition board into the standardized format used by the load simulation, which is shown by the *Hardware Interface* blocks. These hardware interface blocks play a key role in the modularity of the architecture since they allow different hardware to be used without significant changes to the simulation. The third task of the *Computer Simulation* is to simulate any joints that do not have a corresponding hardware module. In some situations it may be desirable to have one or more joint actuators without a hardware component, for example when the hardware is unavailable, too costly, or simply unnecessary. Then the computer simulation must model the joint and interface directly with the control system, shown in the *Actuator Simulation* and *Control Interface* blocks. This third task makes it possible to utilize the RHILS platform at early stages of the design as well as making it more cost effective to set up tests if only one section of the manipulator is under study.

C. Hardware Emulation Block

The *Hardware Emulation* system consists of separate modules for each joint, and each module interfaces with both the *Control System* and the *Computer Simulation*. These modules are further separated into two parts: a *Test Module*, the joint actuator that is being designed/tested, and a *Load Module*, the load-emulating device that mimics the dynamic loads that would be seen in a real system. The *Test Module* includes not only the real actuator, but also the transmission system, position/speed sensors, and motor drive that would be used in the real manipulator, all of which can lead to significant inaccuracies in a pure computer-based simulation. The *Test Module* interfaces directly with the *Control System*, which controls the motor as if it were part of a physical robot. The *Load Module* is coupled to the output of the transmission system, ideally without the use of a secondary transmission that may introduce unwanted uncertainty in the load emulation mechanism. For the range required by most applications, it was found that *torque motors* can supply the necessary torque directly and have other desirable features including consistent torque at low speeds, low inertia, and proper heat dissipation characteristics. The *Load Module* is controlled through a feedback loop that follows the torque calculated by the *Computer Simulation* block. This torque represents the arm dynamics that must be reflected on each joint actuator to have a genuine simulation of the real system. To emulate the dynamic torque accurately closed-loop control is needed, which requires that the torque generated by the *Load Module* be identified. This is done through a unique installation of the torque sensor as a cantilever support for the torque motor (Martin & Emami, 2008).

D. Control System Block
This block can range from running in software on a standard PC to running on dedicated custom hardware depending on the nature and requirements of the application. It is possible to use the real control system for the robot, since as far as the control system is concerned it is connected to the real actuators in a physical robot. This has significant benefits over running a simulated or modified version of the control system: in many applications intense testing of the final control system is required, which can now begin before the final hardware is complete without building expensive prototypes. On the other hand, when the control system is not the focus of the design the flexibility of this architecture allows any simple controller to be quickly implemented and used.

4. LM-RHILS Based Concurrent Engineering of Robot Manipulators

In this section, the *LM* methodology along with the RHILS platform are implemented for building a framework to concurrently design kinematic, dynamic and control parameters of robot manipulators. This framework includes various phases of *LM*, and the *RHILS* is used to evaluate the design attributes and performance supercriterion.

4.1 Architecture

The architecture of the concurrent design framework consists of two parallel workstations, namely *Host* and *Target*, and physical components of a robot manipulator, i.e., three physical joint modules and a controller unit. For each joint module a load emulator is employed to apply simulated dynamic loads during the real-time execution. The collection of load emulators, joint modules and control system is called *Hardware Emulation* block. The entire design architecture and the real physical joint modules are shown in Fig. 3. Although the concurrent engineering framework discussed here is generic and can be applied to any robot manipulator, the *CRS CataLyst-5* manipulator is used in the following implementations for further illustration.

A. Host Workstation
The Host computer is the link between the system and the engineer(s). All design preferences and options are set in this block, where the main code that governs the design process is executed. The preferences are reflected in the satisfactions defined on the design attributes, and the simulation options include initial configuration, the predefined end-effector trajectories, gravity, payload, and the simulation time. This block communicates with the controller to load control gains through an FTP connection, and sends the command signals to the trajectory planner using Python® software. It also loads the kinematic and dynamic parameters and inverse dynamic model of a design candidate to the Target workstation via a TCP/IP connection, and gathers position and torque data that are saved on the Target PC using MATLAB® xPC Target® toolbox. The data are processed and the design attributes are calculated by the Host computer, and considering the design availabilities, the satisfactions are assigned to the design variables and attributes. According to the *LM* methodology, the overall satisfaction of the design candidate is calculated and it is maximized using the MATLAB® optimization toolbox. The optimization of the performance supercriterion is also carried out on the Host computer.

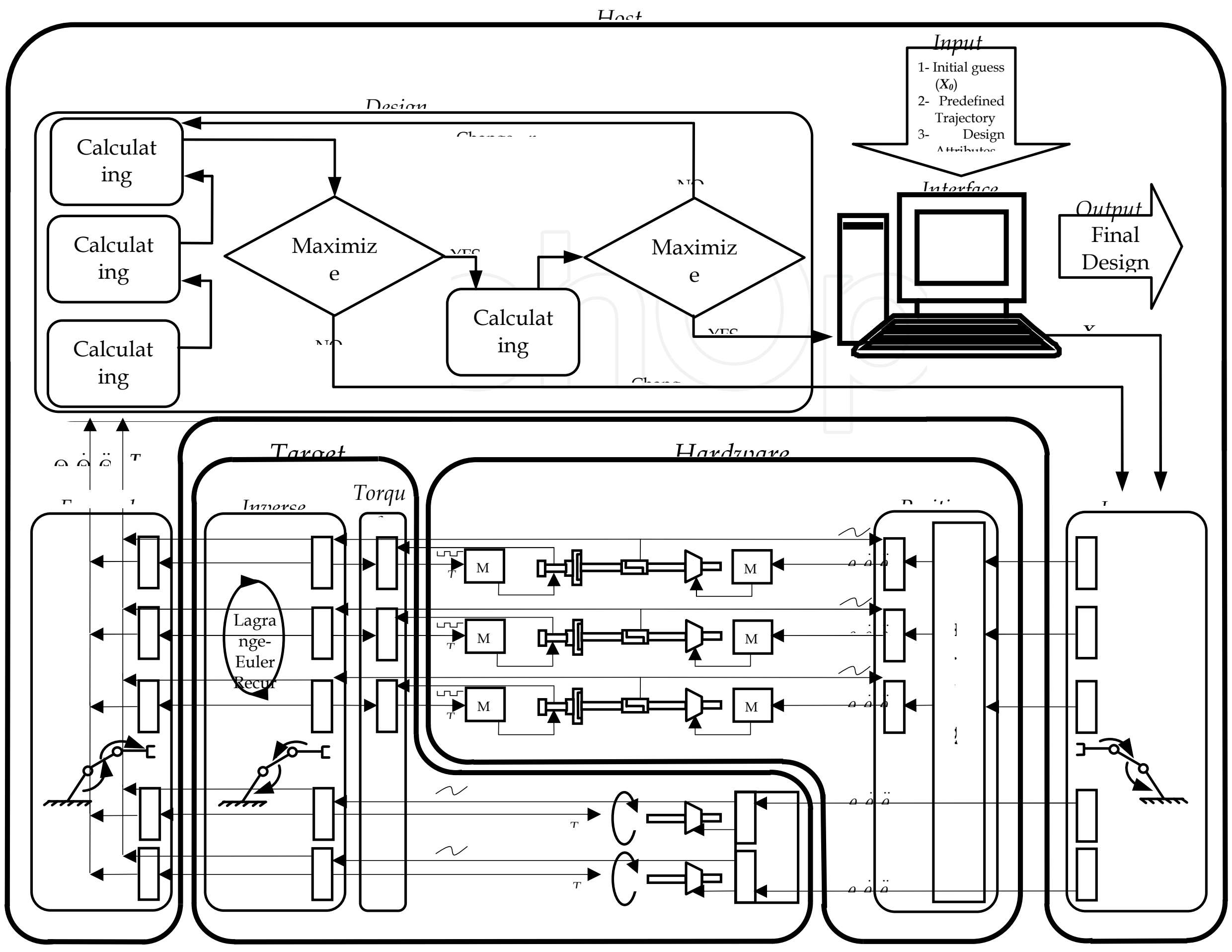


Fig. 3. The LM-RHILS concurrent design architecture

B. Target Workstation

This block is a barebones PC running the xPC Target® real time kernel. On this workstation a servo torque controller for the load emulators and an inverse dynamics model of the manipulator, built in Simulink® and compiled through Real-Time Workshop®, are executed. In the dynamics model, torque signals are calculated based on the kinematics and dynamics of the candidate manipulator and the joints position, velocity and acceleration. The Target computer contains several interface boards to communicate with the joint modules and load emulators. Furthermore, to gather data from the hardware components a data acquisition board and an RS232 port are utilized

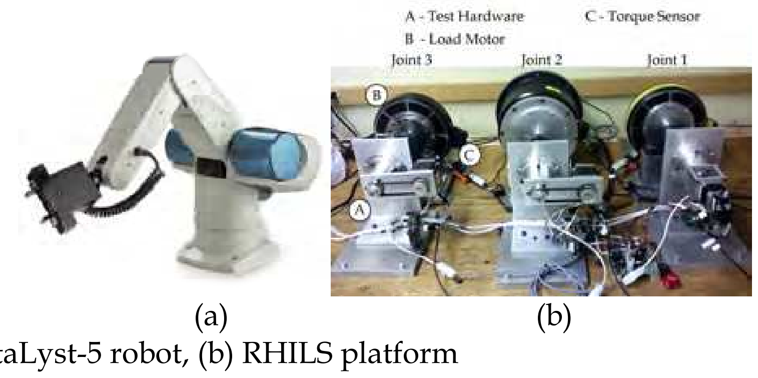


Fig. 4. (a) CRS CataLyst-5 robot, (b) RHILS platform

C. Hardware Emulation

All physical pieces that remain unchanged in the design process form the Hardware Emulation block. Industrial manipulators often have 5 or 6 degrees of freedom (d.o.f.). The first three joints are often used to position the end-effector and the last joints help the wrist change its orientation. Since the first three links are more massive, more force or torque is applied on the corresponding joints, and they play a crucial role in the serial link manipulator performance. Hence, in the design architecture, the first three joint modules of *CRS CataLyst-5* are physically included as a part of the *RHILS* platform, and the rest of the joints are virtually modeled on the Target computer. The corresponding load emulators are also coupled to the joints and the *CRS DM* Master Controller unit is used to control the joint positions. Each joint module consists of a stepper motor, an encoder mounted on the motor shaft, a harmonic drive as a transmission mechanism, and the driver unit. The module interfaces with both the controller and Target workstation in order to receive control signals via motor driver and send joint position to the Target workstation.

The load emulators are coupled directly to the joint shafts to apply the computed loads. These torque signals represent the arm's dynamics and weight and payload effects that must be reflected on each joint actuator to have a genuine simulation of the real system. Since the applied torque should be followed accurately, a servo torque controller is designed and calibrated for each load emulator module. A reaction torque sensor is also installed between the load emulator case (stator) and its mounting fixture to measure the feedback signal. Thus, the load emulator module sends and receives the command and feedback torque signals to and from the Target PC where the torque controller is located (Martin & Emami, 2008).

The controller unit includes a trajectory planner and a typical feedback/feedforward controller for each physical joint module. The trajectory planner generates instantaneous desired position signals with a frequency of 1 KHz based on the input of the controller. Joint trajectories are divided into three sections: first, accelerating to the maximum speed with the nominal acceleration of the joint module, second, constant speed motion and finally, decelerating to the final position with the nominal acceleration.

4.2 Manipulator Concurrent Design Process

In this section, the design architecture is employed to concurrently redesign kinematic, dynamic and control parameters of *CRS-CataLyst-5*. This industrial manipulator consists of 5 rotary joints, three of which are included in the *RHILS* platform. Fig. 4 shows the *CRS-CataLyst-5 manipulator* next to its *RHILS* platform.

In general, the *LM* design framework can be divided into five steps: **a)** decision about design variables and attributes, **b)** assignment of satisfactions, **c)** the primary phase, **d)** the secondary phase, and **e)** the performance supercriterion. However, in this case study, since the existing design is modified and the process can be safely started from the current configuration, the primary phase is not required.

A. Design Variables and Attributes

The kinematic characteristics of a manipulator can be represented by the standard Denavit-Hartenberg convention. Therefore, length (l_i), offset (d_i) and twist (a_i) are considered as kinematic design variables of the i^{th} link. In order to take into account dynamic parameters of the robot, each link is considered as an L-shaped circular cylinder along the link length and offset. The radius of such cylinder (r_i), as a design variable, specifies dynamic parameters of the ith link knowing the link density. The *CRS DM* Master Controller unit generates control signals for each joint consisting of proportional (P_i) and integral (I_i) gains along with gains for feedback velocity ($Kv_{fb,i}$) and acceleration ($Ka_{fb,i}$) and also feedforward velocity ($Kv_{ff,i}$) and acceleration ($Ka_{ff,i}$). Consequently, the design problem deals with $10 \times ndof$ design variables, where *ndof* is the number of degrees of freedom, to identify the most desirable kinematic, dynamic, and control configuration of the manipulator. In the case of *CRS CataLyst-5*, since the last two joints are small at the tip of the manipulator with much less moments of inertia than that of the other joints, their control gains are not considered in the design. Consequently, the design problem deals with thirty-eight design variables in total.

In *LM*, design attributes are divided into *must* and *wish* attributes. The following *must* design attributes are considered:

Design availabilities: Each design variable has an acceptable range of values, considering its physical nature and manufacturing constraints. They are taken into account by the following inequality expression.

$$X_j^{\min} \le X_j \le X_j^{\max} \quad (j = 1,...,n)\,; \tag{38}$$

where $X_j^{\min}$ and $X_j^{\max}$ are the minimum and maximum values for X_j, respectively.

Joint constraint: Since real joint modules are used in the design process, the motor constraints are considered automatically; however, the joints displacements are restricted due to the shape and location of links. This constraint is checked at kth working point for the ith joint angle (θ_i^k) by means of an inequality.

Torque constraint: Each joint module can handle a maximum amount of torque ($\tau_i^{\max}$), usually corresponding to the stall torque of the ith joint motor. Therefore,

$$_{\max}\left|\tau_i\right|^k \le \tau_i^{\max} \quad (i = 1,..., ndof\,; k = 1,..., N)\,; \tag{39}$$

where $_{\max}\left|\tau_i\right|^k$ is the ith joint maximum absolute value of the torque between kth and (*k-1*)th working points.

Maximum reachability: The farthest point that the manipulator can reach is the maximum reachability of the robot (*R*) and because of environmental constraints it should not exceed a certain number (R^{max}).

The main mission of a robot is reflected in the *wish* attributes. In this research, the following *wish* attributes are deemed as the design objectives.

End-effector error: The typical ultimate task for a robot manipulator is to follow predefined trajectories. Therefore, the measured error at the working points is an appropriate *wish* attribute to minimize. If Δ_{tk} and δ_{tk} are the maximum permitted errors for the end-effector position and orientation, respectively, at the kth working point of the tth trajectory, then the end-effector error can be defined as:

$$E = \frac{1}{NT}\sum_{t=1}^{T}\sum_{k=1}^{N}\left(\frac{\sqrt{\Delta x_{tk}{}^2 + \Delta y_{tk}{}^2 + \Delta z_{tk}{}^2}}{\Delta_{tk}} + \frac{\sqrt{\delta x_{tk}{}^2 + \delta y_{tk}{}^2 + \delta z_{tk}{}^2}}{\delta_{tk}}\right) \tag{40}$$

where Δx_{tk} , Δy_{tk} and Δz_{tk} are the position errors in *x, y* and *z* directions, δx_{tk} , δy_{tk} and δz_{tk} are the orientation errors about *x, y* and *z* axes at the k^{th} working point of the t^{th} trajectory, and *T* is the number of trajectories. Note that orientation errors are assumed to be sufficiently small so that the overall orientation error can be considered as a vector. Also, for the 5 d.o.f. *CataLyst-5* manipulator only yaw and roll angles of the end-effector were considered. A maximum of 1*mm* for the translational error and 6° for the orientation error are assigned for this design.

Manipulability: The manipulability index is used for checking the manipulator singularity at the working points. This measure can be expressed as (Bi et al., 1997):

$$M = \frac{1}{N}\sum_{k=1}^{N} cond(J_k{}^0)\ ; \tag{41}$$

where $cond(J_k{}^0)$ is the condition number of Jacobian matrix with respect to the base frame at k^{th} working point. At the singular points the manipulability index approaches infinity and its minimum value is one. Therefore, this *wish* attribute is satisfied when manipulability index is close enough to one.

Structural length index: A desirable manipulator is the one with a smaller Structural length index,

$$Q_L = \left(\sum_{i=1}^{ndof}(l_i + d_i)\right) / \sqrt[3]{V}\ ; \tag{42}$$

where *V* is the workspace volume that can be numerically calculated based on a method detailed in (Ceccarelli et al., 2006).

Total required torque: The total required torque at the k^{th} working point, expressed in (43), can be considered as another *wish* attribute that should be minimized.

$$\tau_T{}^k = \sum_{i=1}^{ndof}\left|\tau_i{}^k\right|\ ; \tag{43}$$

where $\tau_i{}^k$ is the torque of joint *i* at the k^{th} working point.

B. Satisfactions Assignment

Satisfactions are defined as fuzzy membership functions over the range of values that design variables and attributes can obtain. The availability constraints and *must* attributes often satisfy inequalities, while *wish* attributes should be as satisfactory as possible. Since *LM* methodology employs fuzzy set theory, by redefining the notions of inequality and optimization, their restricted binary behaviour can be turned into a flexible and fuzzy one. This brings subjective aspects of design into the scope; in addition, simplifies the design process. One of the popular fuzzy membership functions is the trapezoidal membership function. This function possesses four parameters, i.e., four corners of the trapezoid that the designer should decide about to specify the range in which the satisfaction is one and the slopes of the sides. This decision is made considering the design requirements and the designer's preferences. In other words, the trapezoidal parameters reflect how conservative or aggressive the designer is in interpreting the design attributes. The trapezoids, which are used in this case study, are depicted in Fig. 5. The first and last points of a *must* satisfaction mapping are the minimum and maximum values of the corresponding inequality, respectively. The middle points are picked in a manner that the definition of the inequality is neither too fuzzy nor too crisp, and it obeys the design requirements. For a *wish* satisfaction mapping, the last point is the maximum allowed value of the attribute (for an attribute approaching a minimum), and as it decreases the corresponding satisfaction approaches to one. The middle point is selected based on designer's consensus of the notion of minimum. All minimum and maximum values of design variables and attributes are listed in Table I. Note that since this design problem starts with an existing manipulator configuration and the simulation platform is sufficiently accurate, strict parameters are chosen for defining *wish* satisfactions. This indicates smaller middle ranges and, hence, less steep trapezoid sides.

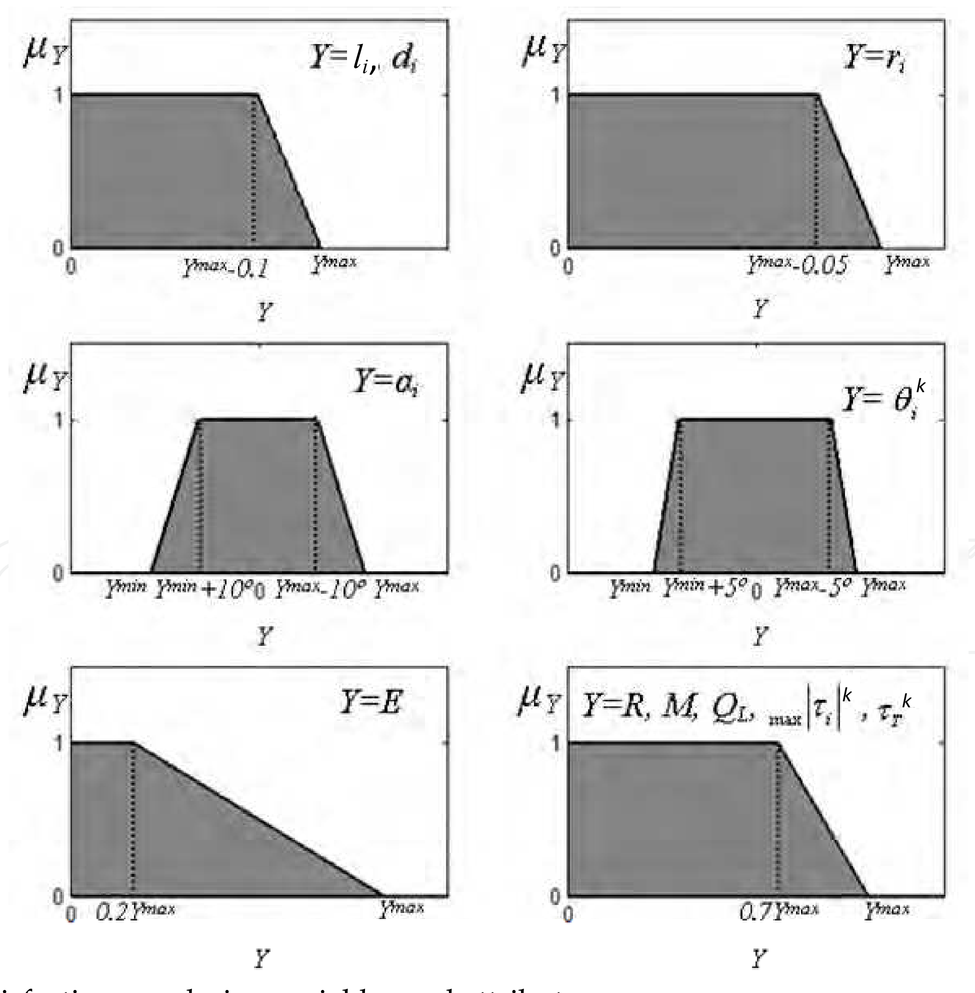


Fig. 5. Satisfactions on design variables and attributes

C. Secondary Phase

To calculate the overall satisfaction, design attributes are determined utilizing the *RHILS* platform that simulates the candidate configuration while it follows a predefined pick-and-place trajectory. In this procedure, first the Denavit-Hartenberg table and dynamic parameters of the design candidate are determined based on the kinematic parameters and the links radii. They are loaded onto the Target workstation as the parameters of the inverse dynamic model of the manipulator. The control gains are also loaded on the controller. On the Host computer an inverse kinematic code is executed to transform the end-effector trajectory to the joint trajectories. The corresponding command signals are sent to the controller from the Host workstation using Python® software and simultaneously, while the real joint modules are moving the joint torques calculated in the Target PC are applied on them by means of the load emulators. Subsequently, the position and torque signals are saved on the Target workstation for further computations. On the Host PC, the design availability, maximum reachability, manipulability and structural length index attributes are calculated using the kinematic parameters. And the joint restriction, torque restriction and total torque required design attributes are determined based on the saved position and torque signals. In addition, a forward kinematic code is executed to compute the actual end-effector position at the working points in order to evaluate the end-effector error. Finally, the corresponding satisfactions are identified and aggregated using the attitude parameters.

The secondary phase searches for the design variables that maximize the overall design satisfaction. A function in the optimization toolbox of MATLAB®, called *fminsearch*, has been employed to perform this single-objective maximization. This function uses a derivative-free search algorithm based on the simplex method that is suitable for handling discontinuity, sharp corners and noise in the objective function, which is the case in this problem. This real-time process takes almost one minute for evaluating each configuration.

i	1	2	3	4	5
$r_i(mm)$	[0,200]	[0,200]	[0,200]	[0,200]	[0,200]
$l_i(mm)$	[0,500]	[0,500]	[0,500]	[0,500]	[0,500]
$d_i(mm)$	[0,500]	[0,500]	[0,500]	[0,500]	[0,500]
$\alpha_i(o)$	[-180,180]	[-180,180]	[-180,180]	[-180,180]	[-180,180]
$\theta_i(o)$	[-180,180]	[-110,0]	[-90.6,35]	[-110,110]	[-180,180]
$_{\max}\lvert\tau_i\rvert(N.m)$	[0,13.8]	[0,13.8]	[0,13.8]	[0,4.8]	[0,2.4]
$R(m)$			[0,0.87]		
E			[0,2]		
M			[1,24]		
Q_L			[0,1.6]		
$\tau_T(N.m)$			[0,12.5]		
Control Gains			$(-\infty,+\infty)$		

Table 1. - Design Variables and Attributes and their Range

D. Performance Supercriterion
By altering the designer's attitude parameters (*p, q* and *a*) the secondary phase generates a set of optimally satisfactory solutions for design. The physical performance of the system should also be checked against an objective *supercriterion*, which is selected to be the total energy consumption at the joints, in order to adjust the designer's attitude.

$$Energy(\boldsymbol{X_s};p,q,\alpha)=\sum_{i=1}^{ndof}\int_{\theta_i^1}^{\theta_i^N}\left|\tau_i d\theta_i\right|; \tag{44}$$

where θ_i^k is the ith joint angle at the kth working point and τ_i is the torque at the ith joint. Ultimately, by minimizing this criterion over optimally satisfactory solutions set ($\boldsymbol{C_S}$), the best design ($\boldsymbol{X}^*$) is achieved.

$$Energy(\boldsymbol{X}^*)=\min_{X_S\in C_S}(\,Energy(\boldsymbol{X_S};p,q,\alpha)). \tag{45}$$

4.3 Some Results and Discussions

The *CRS CataLyst-5* manipulator was redesigned according to the LM-RHILS based concurrent methodology, and the results are shown in Table II. With respect to the manipulator dynamic parameters, the mass of link 3 was reduced by 17.5% as a result of decreasing the link radius and length by 10% and 0.7%, respectively. In addition, all other kinematic and dynamic parameters have been modified slightly, which resulted in enhancing the manipulator performance in terms of the error in the end-effector trajectory, manipulator reachability, workspace and manipulability, and total energy consumption. For example the radius of the first and second links has been changed by almost 0.1% and 0.7%, respectively. The length of link 2 and the offset of link 1 have also been altered by 0.1% and 0.4%, respectively. On the other hand, twist angles have remained almost unchanged. Therefore, in terms of dynamic and kinematic design, the third link has been modified considerably.
In addition, since the controller of the existing manipulator was tuned prior to the redesign process, the control gains have made only slight modifications by an average of 0.8%. Even these small changes in the control parameters significantly affected the end-effector error, *E*, which observed in the results. The error in the end-effector trajectory after the redesign process is approximately 78 times less than its initial value. An increase in the level of satisfaction for all other *wish* attributes can be observed from Table II, as well. Therefore, based on the designer's preferences, all the considered attributes have been enhanced. The total *must* satisfaction has improved, which indicates that the new system is far from its performance limits, and hence the new design is more reliable.
The design candidates obtained from the *LM* secondary phase were optimized against an objective *supercriterion,* which is the total consumed energy, through altering attitude parameters. Ultimately, the configuration with the minimum energy consumption was picked as the final design. The energy consumption was improved by 10%. By looking at the variation of designer's attitude parameters during the design process, one realizes that the initial designer's attitude in aggregating *must* satisfactions was appropriate. That is, the value of *p* did not change through the attitude adjustment. However, in aggregating *wish* satisfactions the designer was originally too conservative. Therefore, *q* was decreased by

50% and α was increased by 140%, approximately, through the attitude adjustment. This implies that instead of focusing on the worst *wish* attribute, the designer should equally stress all *wish* design attributes in order to improve the system energy consumption. Overall, the results show that the original designers of the manipulator (prior to the redesign process) could have been more aggressive (optimistic) in the design of *CRS CataLyst-5*.

5. Conclusion

Concurrent engineering is a promising paradigm for the analysis and synthesis of complex, multidisciplinary systems, such as robot manipulators. It brings *synergy* as a direct consequence of utilizing design knowledge from all participating disciplines, while interacting with each other, and offering equal opportunities to them to contribute to each state of design simultaneously. The advantage, however, does not come at no cost; one must deal with highly-complicated mechatronic system models, and handle optimizations with a large set of multidisciplinary objective and constraint functions and a great number of design variables. The compromise seems to be either to simplify the system model to reduce dimensions of the design space, or to give up the transparency of the design process and appeal to parallel computing algorithms. This chapter discussed an alternative methodology that does not imply any of the above compromises. The new methodology makes the system model computations efficient without compromising design transparency, because it uses the physical system components in the simulation loop, next to the computational model of those modules that need to be designed. The robotic hardware-in-the-loop simulation platform enables the designer to take into account some complex phenomena that are difficult to model, yet execute the entire simulation in real-time. Using hardware components in concurrence with the computational model of the modules that are to be designed results in an effective platform for rapid design alterations. Moreover, the new methodology alleviates the optimization complexities of concurrent design, because it employs Linguistic Mechatronics that not only transforms the multi-objective constrained optimization problem into a single-objective unconstrained formulation, but also formalizes subjective notions and brings the linguistic aspects of communication into the design process.

	$r_i(mm)$					$l_i(mm)$				
	i=1	i=2	i=3	i=4	i=5	i=1	i=2	i=3	i=4	i=5
Initial	65.6	27.7	24.1	10.0	10.0	0.0	254.0	254.0	0.0	0.0
Final	65.7	28.0	21.8	10.0	10.0	0.0	253.6	255.9	0.0	0.0
	$d_i(mm)$					a_i(□)				
	i=1	i=2	i=3	i=4	i=5	i=1	i=2	i=3	i=4	i=5
Initial	254.0	0.0	0.0	0.0	0.0	-90.0	0.0	0.0	-90.0	0.0
Final	255.0	0.0	0.0	0.0	0.0	-90.8	0.0	0.0	-90.7	0.0
	P_i			I_i			$Kv_{fb,i}$			$[p,q,a]$
	i=1	i=2	i=3	i=1	i=2	i=3	i=1	i=2	i=3	
Initial	18.32	20.00	12.00	0.073	0.050	0.100	40.7	40.0	20.0	[10,1.5,0.5]
Final	18.46	20.16	12.10	0.074	0.050	0.101	41.0	40.3	20.2	[10,0.7,1.2]
	$Ka_{fb,i}$			$Kv_{ff,i}$			$Ka_{ff,i}$			*Energy (J)*
	i=1	i=2	i=3	i=1	i=2	i=3	i=1	i=2	i=3	

Initial	43.4	100.0	80.0	59.0	40.0	30.0	3473.0	100.0	120.0	6.2549
Final	43.8	100.8	80.6	59.5	40.3	30.2	3483.6	100.8	120.9	5.6307
	Wish Design Attributes									
	E	M	Q_L	$\tau_T^k\,(N.m)$						
				k=1	k=2	k=3	k=4	k=5	k=6	
Initial	1.4787	20.7223	1.3091	9.3557	10.2754	9.3561	9.3561	10.2172	10.2172	
Final	0.0189	19.4921	1.3025	8.3071	9.1391	8.3071	8.3071	9.1394	8.3071	
	Wish Satisfactions									Overall Satisfaction
	μ_E	μ_M	μ_{Q_L}	$\mu_{\tau_T^k}$						
				k=1	k=2	k=3	k=4	k=5	k=6	μ
Initial	0.000	0.606	0.455	0.838	0.593	0.838	0.838	0.609	0.609	0.250
Final	1.000	0.620	0.626	1.000	0.896	1.000	1.000	0.896	1.000	0.607

Table 2. – Results of Concurrent Design

The new methodology of concurrent engineering was used to redesign the kinematic, dynamic, and control parameters of an industrial manipulator, namely *CRS CataLyst-5*, whose joint modules had been installed in the *RHILS* platform. Despite the fact that the existing manipulator design had been well developed, the new design enhanced the system performance (end-effector trajectory error, manipulator reachability, workspace and manipulability, and total energy consumption) by changing the current manipulator configuration.

6. References

Ballard, B. L., Elwell Jr., R. E., Gettier, R. C., Horan, F. P., Krummenoehl, A. F. and Schepleng, D. B. (2002). Simulation Approaches for Supporting Tactical System Development, *John Hopkins APL Technical Digest* (*Applied Physics Laboratory*), Vol. 23, No. 2-3, pp. 311-324.

Bi, Z. M., Li, Y. F. and Zhang, W. J (1997). A New Method For Dimensional Synthesis of Robotic Manipulators, *5th National Applied Mechanisms and Robotics Conference,* Cincinnati.

Bi, Z. M. and Zhang, W. J. (2001). Concurrent Optimal Design of Modular Robotic Configuration, *Journal of Robotic Systems,* Vol. 18, No. 2.

Borutzky, W. (2006). Bond Graph Modeling and Simulation of Mechatronic Systems: An Introduction into the Methodology, *Proceeding 20th European Conference on Modeling and Simulation (ECMS)*.

Breedveld, P. C. (2004). Port-based Modeling of Mechatronic Systems, *Mathematics and Computers in Simulation*, Vol. 66, pp. 99-127.

Castano, A., Behar, A. and Will, P. M. (2002). The Conro modules for reconfigurable robots, *IEEE/ASME Transactions on Mechatronics*, Vol. 7, No. 4, pp. 403-409.

Ceccarelli, M., Carbone, G. and Ottaviano, E. (2005). An Optimization Problem Approach for Designing Both Serial And Parallel Manipulators, *The International Symposium on Multibody Systems and Mechatronics Proceedings of MUSME,* Brazil.

Chhabra, R. and Emami, M.R. (2009). Design Criteria in Mechatronics, submitted to *IEEE/ASME Transactions on Mechatronics*, September 2009.

Chirikjian, G. S. (1994). Kinematics of a Metamorphic System, *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 449-455.

Coello, C.A. (1999). A Comprehensive Survey of Evolutionary Based Multiobjective Optimization Techniques, *Knowledge and Information Systems An International Journal*, 1(3): 269–308.

Cyril, X., Jaar, G. and St-Pierre, J. (2000). Advanced Space Robotics Simulation for Training and Operations, *AIAA Modeling and Simulation Technologies Conference*, pp. 1-6, Denver, USA.

Dhingra, A. K., Rao, S. S. and Miura, H. (1990). Multiobjective Decision Making in a Fuzzy Environment with Application to Helicopter Design, *AIAA Journal,* Vol. 28, No. 4.

Dhingra, A. K. and Rao, S. S. (1995). A Cooperative Fuzzy Game Theoretic Approach to Multiple Objective Design Optimization, *European Journal of Operational Research,* Vol. 83, pp. 547-567.

Emami, M. R. (1997). *Systematic Methodology of Fuzzy-logic Modeling and Control and Application to Robotics,* Ph.D. Dissertation, Department of Mechanical and Industrial Engineering, University of Toronto, Canada.

Emami, M. R., Turksen, I.B., Goldenberg, A.A. (1998). Development of a systematic methodology of fuzzy-logic modeling, *IEEE Trans. Fuzzy Systems*, Vol. 6, No. 3, pp. 346-361.

Emami, M. R., Türksen, I. B. and Goldenberg, A. A. (1999). A Unified Parameterized Formulation of Reasoning in Fuzzy Modeling and Control, *Fuzzy Sets and Systems,* Vol. 108, pp. 59-81.

Hanselman, H. (1996). Hardware-in-the-loop Simulation Testing and its Integration into a CACSD Toolset, *IEEE International Symposium on Computer-Aided Control System Design*, pp. 15-18.

Hewit, J. (1996). Mechatronics design – the key to performance enhancement, *Robotics and Autonomous Systems*, No. 19, pp. 135-142.

Hu, X. (2005). Applying Robot-in-the-loop Simulation to Mobile Robot Systems, *12th International Conference on Advanced Robotics (ICAR 2005).*

Kurapati, A., Azarm, S. and Wu, J. (2000). Constraint Handling in Multiobjective Genetic Algorithms, In: *Proceedings of 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, Paper No. AIAA-2000-4893.

Leitner, J. (1996). Space Technology Transition Using Hardware in the Loop Simulation, *Proceedings of the 1996 Aerospace Applications Conference*, Vol. 2 , pp. 303-311.

Li, C.-J. and Sankar, T. S. (1992). Fast inverse dynamics computation in real-time robot control, *Mechanism & Machine Theory*, Vol. 27, No. 6, pp. 741-750.

Linjama, M., Virvalo, T., Gustafsson, J., Lintula, J., Aaltonen, V. and Kivikoski, M. (2000). Hardware-in-the-loop Environment for Servo System Controller Design, Tuning, and Testing, *Microprocessors and Microsystems*, Vol. 24, No. 1, pp. 13-21.

Maclay, D. (1997). Simulation gets into the loop, *IEE Review*, Vol. 43, No. 3, pp. 109-112.

Martin, A. and Emami, M. R. (2008). Design and Simulation of Robot Manipulators using a Modular Hardware-in-the-loop Platform, in M. Ceccarelli (ed.) *Robot Manipulators: Programming, Design, and Control,* I-Tech Education and Publishing, Vienna, Austria, pp. 347-372.

Murata, S., Yoshida, E., Tomita, K., Kurokawa, H., Kamimura, A. and Kokaji, S. (2000). Hardware Design of Modular Robotic System, *Proceedings of the 2000 IEEE International Conference on Intelligent Robots and Systems,* pp. 2210-2217.

Otto, K. N. and Antonsson, E. K. (1995). Imprecision in Engineering Design, *ASME Journal of Mechanical Design,* Vol. 117(B), pp. 25-32.

Paredis, C.J.J. (1996). *An Agent-based Approach to the Design of Rapidly Deployable Fault Tolerant Manipulators*, PhD Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA.

Paynter, H. M. (1961). Analysis and Design of Engineering Systems, M.I.T. Press, Cambridge, Massachusetts, USA.

Piedboeuf, J.-C., Carufel, J. de, Aghili, F. and Dupuis, E. (1999). Task Verification Facility for the Canadian Special Purpose Dexterous Manipulator, *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pp. 1077–1083, USA.

Ramachandran, S. and Chen, I.-M. (2000). Distributed Agent Based Design of Modular Reconfigurable Robots, *Proceeding of the 5th International Conference on Computer Integrated Manufacturing*, Singapore, pp. 447–458.

Rus, D. and McGray, C. (1998). Self-Reconfigurable Modular As 3-D Metamorphic Robots, *Proceedings of the 1998 IEEE International Conference on Intelligent Robots and Systems,* pp.837-842.

Rus, D. and Vona, M. (2000). A Physical Implementation of Self-Reconfigurable Crystalline Robot, *Proceedings of the 2000 IEEE International Conference on Robotics and Automation,* pp. 1726-1733.

Stoeppler, G., Menzel, T. and Douglas, S. (2005). Hardware-in-the-loop simulation of machine tools and manufacturing systems, *Computing & Control Engineering Journal*, Vol. 16, No. 1, pp. 10-15.

Temeltas, H.; Gokasan, M., Bogosyan, S. and Kilic, A. (2002). Hardware in the Loop Simulation of Robot Manipulators through Internet in Mechatronics Education, *The 28th Annual Conference of the IEEE Industrial Electronics Society*, Vol. 4, pp. 2617-2622, Sevilla, Spain.

Thermo Fisher Scientific Inc. (2007). CRS CataLyst-5 Robot System, Website: http://www.thermo.com/com/cda/product/detail/0,1055,21388,00.html.

Yager, R. R. and Filev, D. P. (1994). Essentials of Fuzzy Modeling and Control, *New York: John Wiley and Sons*.

Yoshida, E., Kokaji, S., Murata, S., Kurokawa H. and Tomita, K. (1999). Miniaturised Self-Reconfigurable System Using Shape Memory Alloy, *Proceedings of the 1999 IEEE International Conference on Intelligent Robots and Systems,* pp. 1579-1585.