

CPS Modeling Integration Hub and Design Space Exploration with Application to Microrobotics

Yuchen Zhou and John S. Baras

The Institute for Systems Research
and Electrical and Computer Engineering Department,
University of Maryland, College Park, Maryland, USA
{yzh89, baras}@umd.edu

Abstract. We describe a new methodology and environment for Cyber Physical Systems (CPS) synthesis and demonstrate it in the design of microrobots viewed as CPS. Various types of microrobots have been developed in recent years for applications related to collaborative motion such as, sensor networks, exploration and search-rescue in hazardous environments and medical drug delivery. However, control algorithms for these prototypes are very limited. Our new approach for modeling and simulation of the complete microrobotics system allows the robots to complete more complex tasks as per specifications. Since the microrobots tend to have small features, complex micro-structures and hierarchy, the control laws cannot be designed separately from the physical layer of the robots. Such a type of microrobot is indeed a CPS, as control in the cyber side, and the material properties and geometric structure in the physical side, are tightly interrelated. This design approach is important for microrobots, capable of collaborating and completing complex tasks.

Keywords: Modelica, Microrobot, CPS, System model.

1 Introduction: Synthesis Environment for CPS

The rapid development of information technology (in terms of processing power, embedded hardware and software systems, comprehensive IT management systems, networking and Internet growth, system design and integration environments) is producing an increasing number of applications and opening new doors. In addition, over the last decade, we entered a new era where systems complexity has increased dramatically. Complexity is increased both by the number of components that are included in each system as well as by the dependencies between those components. Cyber-Physical Systems (CPS) are *engineered systems* constructed as networked interactions of physical and computational (cyber) components. In CPS, computations and communications are deeply embedded in and interacting with physical processes, and add new capabilities to physical systems. The challenge in CPS is to incorporate the inputs (and their characteristics and constraints) from the physical components in the logic of the

cyber components (hardware and software). Whole industrial sectors are transformed by new product lines that are CPS-based. Modern CPS are not simply the connection of two different kinds of components engineered by means of distinct design technology, but rather, a new system category that is both physical and computational. Current industrial experience tells us that, in fact, we have reached the limits of our knowledge of how to combine computers and physical systems. The shortcomings range from technical limitations in the foundations of cyber-physical systems to the way we organize our industries and educate engineers and scientists that support cyber-physical system design. If we continue to build systems using our very limited methods and tools but lack the science and technology foundations, we will create significant risks, produce failures and lead to loss of market.

If a successful contribution is to be made in shaping this change, the revolutionary potential of CPS must be recognized and incorporated into internal development processes at an early stage. For that *Interoperability and Integratability* of CPS is critical. In our recent research [1], [2], [3], [4], [5], [6] we have initiated the development of a framework to facilitate interoperability and integratability of CPS. Currently there is a lack of well-defined tools and synthesis environments for CPS. CPS synthesis requires cross-domain concepts for architecture, communication and compatibility at all levels. The effects of these factors on existing or yet undeveloped systems and architectures represent a major challenge. The aim of our recent research is precisely to clarify these objectives and systematically develop detailed recommendations and synthesis environments for CPS. We have focused our efforts in two essential problems:

- (i) A framework for developing cross-domain *integrated modeling hubs* for CPS.
- (ii) The creation and demonstration of an initial framework for linking the integrated CPS modeling hub of (i) with powerful and diverse *tradeoff analysis methods* and tools for *design space exploration* for CPS.

1.1 Model-Based Systems Engineering

MBSE [7] has emerged as a promising methodology for the systematic design, performance evaluation and validation of complex engineering systems. “Model-Based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases” [7]. MBSE facilitates the flow of requirements through models, a methodology that is at the same time compact and enforces consistency between data and requirements (through the models). Figure 1 describes the basic steps of the MBSE process that we have developed, and have been teaching at the University of Maryland (UMD) for several years. A most recent development of particular importance is the development and teaching of a new hands-on undergraduate course at UMD, *ENES489P “Hands-on Systems Engineering Projects”*. This MBSE process has the following steps (phases): Requirements Collection, Construction of System Structure Model (what the system consists of), Construction of System Behavior Model (what the system

does), Mapping of Behavior onto Structure (what structure components will perform parts of behavior), Allocation of Requirements to Structure and Behavior Components, Trade-Off Analysis, Validation and Verification. As illustrated in Figure 1, the process moves between these steps in an iterative manner, until satisfactory alternative system designs are developed. The process is executed at different levels of granularity (detail/aggregation). As the MBSE process executes a *system architecture* is developed through the creation of behavior and structure components, their interrelationships and the allocation of behavior components to structure components.

1.2 Systems Modeling Language (SysML)

SysML [8] is a general purpose graphical modeling language that was developed based on UML and is a key enabler for the MBSE process by providing ways for the representation and analysis of complex engineering systems. SysML supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data,

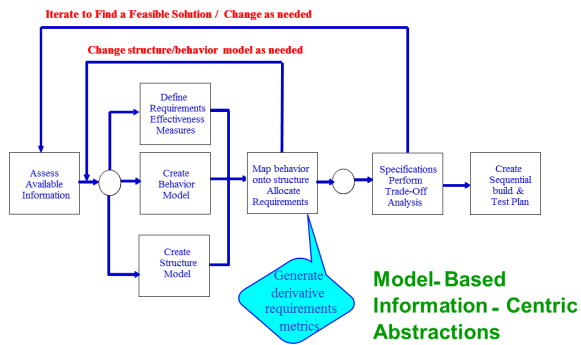


Fig. 1. Model-Based Systems Engineering Process [9]

personnel, procedures, and facilities. SysML supports model and data interchange via XML Metadata Interchange (XMI) and the AP233 standard. Recent research has demonstrated the use of SysML [8] as a centerpiece abstraction for team-based system development, with a variety of interfaces and relationship types (e.g., parametric, logical and dependency) providing linkages to detailed discipline-specific analyses and orchestration of system engineering activities. The four fundamental pillars of SysML are the support of models for the *structure* of the system, models of the *behavior* of the system, models for capturing the requirements for the system via the new *requirements diagram* of the system, and the new and innovative *parametric diagram* of the system, which ties design variables and metric parametric representations to the structure and behavior models (a kind of annotation of these models). Parametric diagrams are the key to linking SysML-based system models to analysis models, including trade-off analysis models such as multi-metric optimization (e.g. IBM-ILOG CPLEX) and constraint based reasoning tools (e.g. IBM-ILOG Solver). SysML, as a language for describing the system architecture, is a catalyst for the integration of various modeling environments, as well as analysis/design environments, for complex systems, while allowing multiple disciplinary views of the system and its components, as illustrated in Figure 2, where the System Architecture Model

is described via SysML. Our research has taken several key steps towards the development of new foundations for this model integration framework we call **CPS modeling integration hubs**. We have recently developed [3], [6] such hubs for power grids, microrobotics, energy efficient buildings, vehicle management systems for next generation all-electric aircraft, sensor networks, robotics and collaborative swarms.

1.3 CPS Modeling Integration Hub Architecture

A major challenge in MBSE for CPS is to have models that are consistent with each other. However, besides having consistent data there is a need for the models to work together in order to offer a holistic Systems Engineering approach to the designer of CPS. SysML is used in the core of our modeling integration hub (Fig. 2 and Fig. 3). The main aim is to integrate this core module with external tools, each one used in a different phase of the Systems Engineering process [10]. The resulting MBSE environment can be thought of as a “virtual” product line management (PLM) environment for CPS, across discipline tools. To achieve this integration a three-layer approach needs to be followed. Initially, for the tool we need to integrate, a domain specific profile is created in SysML. Then a model transformation is defined, followed by the implementation of tool adapters that are used as a middleware for exchanging information between the model transformation layer and the other components of the hub. Fig. 3 presents these layers as well as the areas for which we need to integrate tools with the core module to realize the MBSE vision of a system design experience for CPS.

A key component of the emerging framework is a metamodeling environment with its associated languages and its semantics based on sophisticated versions of annotated block diagrams and bond graphs [6]. A metamodeling layer stands one abstraction layer above the actual design implementation in a modeling language. A metamodel consists of the constructs of a modeling language together with the rules that specify the allowable relationships between these constructs. It can be considered as the grammar of that modeling language. At the metamodeling layer model transformations take place. There are many alternatives in terms of model transformation tools, like ATL, GME, eMoflon, QVT. In our research

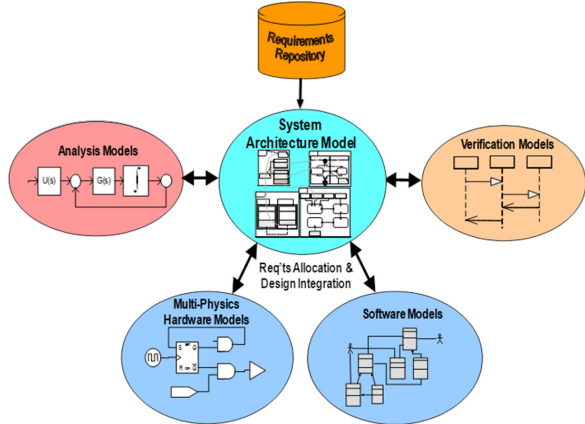


Fig. 2. Multi-domain model integration via system architecture model (SysML)

the eMoflon model transformation tool was used [11], [12]. Finally, tool adapters work as the “glue” between the different pieces of software. Their role is to access/change information inside a model and call the appropriate Java functions generated by the eMoflon tool to perform model transformations [6], [13].

1.4 Tradeoff Analysis and Design Space Exploration

Although progress to date in MBSE facilitates the integration of system component models from different domains, we still need an integrated environment to optimize system architecture, manage the analysis and optimization of diverse measures of effectiveness (MoE), manage the various acceptable designs and most than anything else perform tradeoff analysis. Tradeoff is an essential part of system design, as it implements design space exploration. SysML does not provide a way for engineers to formally evaluate and rank design criteria, conduct sensitivity analysis, search design spaces for better design solutions, and conduct trade studies.

To address this challenge we have introduced [6] the concept that SysML needs to be integrated with industrial-strength multi-objective algorithms, constraint-based reasoning algorithms, with appropriate linkages to modeling/simulation environments. An integration of SysML with a tradeoff tool will allow the designer to make decisions faster and with more confidence.

We have recently developed and demonstrated [6] the first ever integration of a powerful tradeoff analysis tool (and methodology), Consol-Optcad, which is a sophisticated multi-criteria optimization tool developed at the University of Maryland, with our SysML-based modeling integration hubs for CPS. Consol-Optcad is a multi-objective optimization tool that allows interaction between the model and the user. It can handle non-linear objective functions and constraints with continuous values. Another version of Consol-Optcad has been developed to handle also logical variables, via integer and constraint programming [14]. In systems development and after the system structure is defined there is a need to calculate the design parameters that best meet the objectives and constraints. Usually when we deal with complex systems and optimization is under consideration, this is not a trivial task. The support of an interactive tool, like Consol-Optcad, to help the designer resolve the emerging trade-offs is necessary.

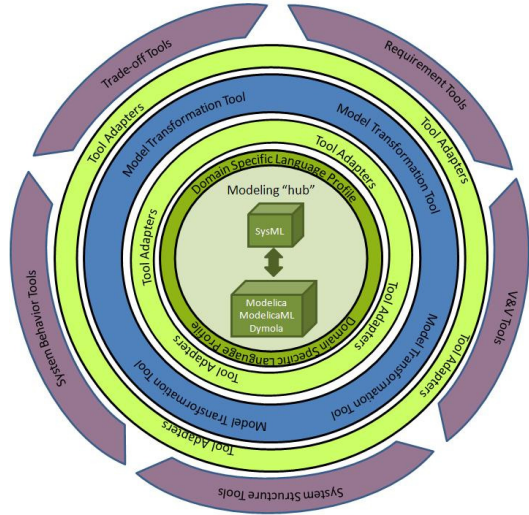


Fig. 3. The Modeling Integration Hub for CPS

A major advantage of Consol-Optcad is that it allows the user to interact with the tool, while the optimization is under way. The designer might not know or might not be in a position at the beginning to specify what preferred design means. Therefore such interaction with the tool could be of great benefit [15], [16]. Another key feature of Consol-Optcad is the use of the Feasible Sequential Quadratic Programming (FSQP) algorithm for the solver. FSQP's advantage is that as soon as we get an iteration solution that is inside the feasible region, feasibility is guaranteed for the following iterations as well. Moreover, very interesting is the fact that besides traditional objectives and constraints Consol-Optcad allows the definition of functional constraints and objectives that depend on a free parameter. Consol-Optcad has been applied to the design of flight control systems [17], rotorcraft systems [18], integrated product process design (IPPD) systems [14] and other complex engineering systems.

2 System Level Design of Microrobots as CPS

Microrobotics have been of particular interest to researchers in Robotics and Control, because of their wide application in collaborative control, medical sensors [19], mobile sensor networks for surveillance [20] and microrobot self-assembly [21]. Many of the recent prototype designs are based on Microelectromechanical systems (MEMS) fabrication processes using specific mechanisms to create planar motion through miniature structures. These include using force from electrical static force [22], thermo bending [23] and chemical reactions [24]. However, current prototype-based design methodology for microrobots is not systematic. In the design process, control policies are normally designed completely separately from the structure after the prototype is manufactured [21], [23]. Such a process either makes it impossible for the robot to accomplish complicated tasks autonomously, or require external force to control the motion of the robot [19]. Modeling the robot requires a very precise description to the physical layers in the process. Material constraints and material properties are critical for the microrobotic design [25]. Complex control laws, which are the cyber side, will not perform well if the physical robot is not well modeled. Moreover, the simulation and design process of the cyber part become increasingly dependent on the physical model and will be directly influenced by any changes made in the physical design. On the other hand, the cyber part affects the stability and controllability of the physical model as well. This makes the microrobot a complicated CPS. Therefore, a model-based systems engineering approach including simulation and validation is needed for this process.

In this paper we follow the methodology described in Sections 1.1-1.4, for the system level modeling and design of microrobots viewed (properly) as CPS. To model the cyber and physical layers, we chose the Modelica language, due to its well-known capability for modeling complex physical systems problems [26].

In this paper we focus on a type of walking robot that uses six legs to alternately support the structure and moving forward similar to that of an insect. Instead of designing a whole new robot, we demonstrate the possible design exploration enabled by our methodology on a particular prototype of a walking

microrobot described in [27]. The subject robot [27] utilizes flexible joints to damp the impact with the ground so as to stabilize the walking motion. Although the overall structure is not complex as shown in Fig. 4a, the model can be easily made unstable due to the ground collisions even with the flexible joints. In order to create a more stable model and further explore the design space, a system level model for this particular robot is created to investigate its stability, structure alternatives and efficiency.

The rest of the paper is organized as follows. Section 3 gives the analytical approach for the physical model of the subject walking robot [27]. Section 4 includes Mod-elica simulation and results related to the stability and planar motion of the robot. Section 5 presents possible material choices and design exploration suggested by the simulation and also its effects on the control laws. Section 6 gives suggestions on tradeoff analysis and validation before prototype fabrication using a model-based system engineering approach.

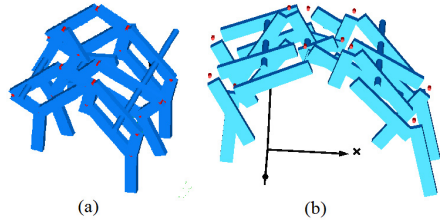


Fig. 4. The original millirobot is on the left and the modified millirobot is on the right

3 The Physical Model

The particular microrobots we are interested in are small robots with micro features, more specifically with flexible joints which make them more stable. In the first part of this section, the mechanism of flexible joints will be discussed and approximated by a torsion spring according to beam theory. In the second part, the rigid part of the microrobot will be analyzed using multibody dynamics and kinematics derived from a Lagrangian formulation. The ground interaction is discussed at the end of this section. The physical model creates constraints for the controller, since the flexible part will break if large force or torque is applied, and instability can easily arise from the poorly designed structure.

3.1 Flexible Joint Model

Assuming small bending of the joint, all flexible joints of the robot are modeled as torsion springs derived from beam theory. According to beam theory, the local curvature ρ of the bending beam is

$$\frac{1}{\rho} = \frac{M}{EI_z}. \quad (1)$$

where M is the applied moment, E is the Young Modulus of the beam and I_z is the inertia about the rotational axis. In the case of a rectangular beam, $I_z = 1/12bh^3$, where b is the beam height and h is the beam width.

For small values of the angle, we obtain a linear relation between the angle and the torque. The spring constant is therefore,

$$k = \frac{EI_z}{l},$$

where l is the beam length.

3.2 Kinematics Model

Consider the kinematics model described in Fig. 5. Let $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2$ be the coordinate frames attached to the joints as shown. Denote $(V_{i/j}^P)_{\mathcal{R}_k}$ as the velocity of point P attached to body i (B_i) relative to body j (B_j) expressed in the coordinate frame \mathcal{R}_k . The twist, including rotational and translational velocities, of P can be described with respect to \mathcal{R}_0 using $\hat{V} = [w \ v]^T$ [28],

$$\left(\hat{V}_{3/0}^P\right)_{\mathcal{R}_3} = \left(\hat{V}_{3/2}^P\right)_{\mathcal{R}_3} + \left(\hat{V}_{2/1}^P\right)_{\mathcal{R}_3} + \left(\hat{V}_{1/0}^P\right)_{\mathcal{R}_3}.$$

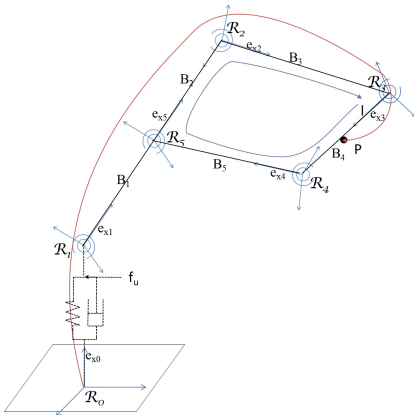


Fig. 5. Mechanical model of one single leg

where $e_{x,y,z}^i$ are unit vectors codirectional with the axes of frame \mathcal{R}_i , for $i = 0, 1, \dots, 5$. All unit vectors of different frames can be expressed in terms of $e_{x,y,z}^3$ in \mathcal{R}_3 alone using coordinate transformations. The Jacobian J of the robot is

$$\left(\hat{V}_{3/0}^P\right)_{\mathcal{R}_3} = J \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{pmatrix}$$

Then, we can express J at the point P on B_4 as,

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ -l_2 \sin \theta_3 - l_1 \sin \theta_2 + \theta_3 & -l_2 \sin \theta_3 & 0 \\ -l - l_2 \cos \theta_3 - l_1 \cos \theta_2 + \theta_3 & -l - l_2 \cos \theta_3 & -l \\ 0 & 0 & 0 \end{pmatrix}$$

Let l_1 be the length of B_1 and B_2 as shown in Fig. 5 (B_1 and B_2 constitute one rigid segment), l_2 be the length of B_3 and l be the distance between the origin of \mathcal{R}_3 and the point of interest P . Then the relative twists are,

$$\left(\hat{V}_{3/2}^P\right)_{\mathcal{R}_3} = \begin{pmatrix} \dot{\theta}_3 e_z^3 \\ l e_x^3 \times \dot{\theta}_3 e_z^3 \end{pmatrix}$$

$$\left(\hat{V}_{2/1}^P\right)_{\mathcal{R}_3} = \begin{pmatrix} \dot{\theta}_2 e_z^2 \\ (l_2 e_x^2 + l e_x^3) \times \dot{\theta}_2 e_z^2 \end{pmatrix}$$

$$\left(\hat{V}_{1/0}^P\right)_{\mathcal{R}_3} = \begin{pmatrix} \dot{\theta}_1 e_z^1 \\ (l_1 e_x^1 + l_2 e_x^2 + l e_x^3) \times \dot{\theta}_1 e_z^1 \end{pmatrix},$$

where $e_{x,y,z}^i$ are unit vectors codirectional with the axes of frame \mathcal{R}_i , for $i = 0, 1, \dots, 5$. All unit vectors of different frames can be expressed in terms of $e_{x,y,z}^3$ in \mathcal{R}_3 alone using coordinate transformations. The Jacobian J of the robot is

The inner loop of the leg in Fig. 5 represents the additional constraint of the system dynamics, which reduces the system degrees of freedom (DOF). Using the same method as above, we can compute J_c which has constraint $0 = J_c \dot{q}$, where q is the vector of the generalized coordinates of the robot with entries θ_i , $i = 1, 2, \dots, 5$. In the leg model this constraint reduces the necessary coordinates from 5 to 4.

To solve the dynamics using Euler-Lagrange equation, the kinetic energy has to be expressed in terms of the generalized coordinates. By using the method above, we can compute the velocity of every interest point P in terms of generalized coordinates θ_1 , θ_2 , θ_3 , and θ_4 . For instance, the kinetic energy of B_1 and B_2 relative to the lab frame is,

$$(T_{1/0})_{\mathcal{R}_0} = \frac{1}{2}(\dot{\theta}_1 e_z^1)I(\dot{\theta}_1 e_z^1) + \frac{1}{2}m_1 \left(\dot{\theta}_1 e_z^1 \times \frac{l_1}{2} e_x^1 \right)^2,$$

which can be expressed in terms of θ_1 and $\dot{\theta}_1$ after performing coordinate transformation on e_x^1 . We apply the same process to express the kinetic energy of B_3 , B_4 , B_5 but with an extra term to incorporate motion of the center of mass.

The conservative forces are gravity force, ground interaction force, and torsion spring tension. Using the same coordinate transformation method, we can express them in generalized coordinates as well.

We can invoke the Euler-Lagrange equation for every leg segment, with Q being the sum of all conservative forces on that segment,

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{q}} - \frac{\partial T}{\partial q} = Q \quad (2)$$

3.3 Ground Interaction

The ground interaction as shown in Fig. 5 is modeled as spring and damper with kinetic friction whenever contact is present. The static friction is not included because the contact time is short and the tangential speed is always not zero. First the state is augmented with a relative vertical distance between the origin of \mathcal{R}_1 and the origin of \mathcal{R}_0 . The kinetic friction and normal force is shown in Equation (2) only when the relative distance is less than zero. There is no rotation in the normal direction of collision, thus no torque.

The normal force f_v and horizontal force f_h is formulated as follows, [29]

$$\begin{aligned} f_v &= \min(ky + dv_n, 0) \\ f_h &= -\mu_m \|f_v\| \begin{cases} 1 & \text{if } \|v_h\| > v_{min} \\ \|v_h\|/v_{min} & \text{else} \end{cases} \end{aligned}$$

where k is the spring constant, d is the damping term, y is the deformation in the vertical direction, v_n is the approaching velocity in the vertical direction, v_h is the relative velocity in the horizontal direction, v_{min} is an adjustment term to avoid abrupt changes in the friction force through sign changes in v_n .

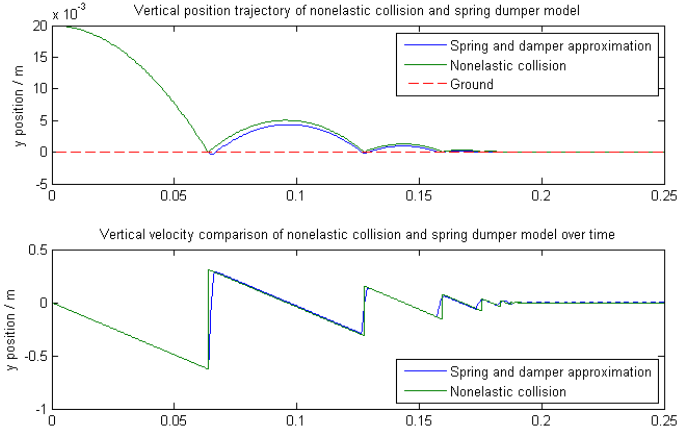


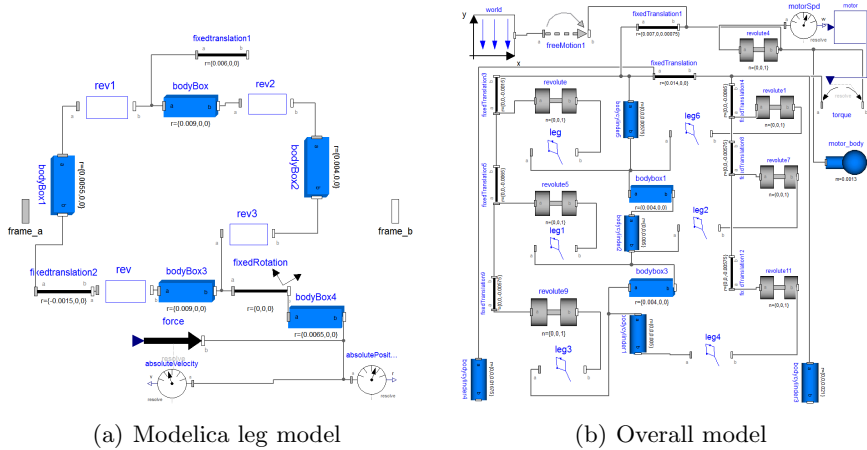
Fig. 6. Simple example of a bouncing ball using spring damper model and nonelastic collision. Upper plot shows y position while bottom one gives the velocity comparison.

This contact model is a model that is linear in distortion and linear in approaching velocity. The coefficient is tuned to achieve similar energy loss with nonelastic collision with coefficient of restitution 0.5. As shown in Fig. 6, the result of the spring and damper model is well fit with the nonelastic collision model with the chosen spring and damper constants. A more precise static model is the Hertz model. In this case, the contact force is proportional to the distortion to the power of $2/3$. The Hertz model is a precise static model and it requires a more precise understanding of the contact point. Because the contact point of the millirobot has different shapes over the simulation time, the simplified spring and damper models are more suitable for ground modeling.

4 Simulation Results and Discussion

The Modelica millirobot model is pieced together according to a Pro Engineering model used for the initial structure design of the subject robot [27]. The leg, as shown in Fig. 5, is modeled in Modelica as in Fig. 7(a). The joints have specific details such as joint length, specified by the designer, and spring constant determined by Equation (1) according to the material properties and geometry of the flexible joint. The leg model is then linked together with other pieces as shown in Fig. 7(b).

In simulation animation, the robot is seen as in Fig. 4. The overall Modelica model of the robot is close to the physical model, with modifiable parameters for geometry and material properties of the joints and rigid body parts. The simulation results show that the model behaves close to the experiment (Fig. 8).



(a) Modelica leg model

(b) Overall model

Fig. 7. Fig. (a) describes the structure of the leg model in Modelica block diagram. The joints rev, rev1, rev2 and rev3 are the joints with flexible material. Fig. (b) gives a simplified structure of the robot using the leg submodel. The shaft mechanism is in the middle with linkages to six legs on either side. The top right portion depicts the motor.

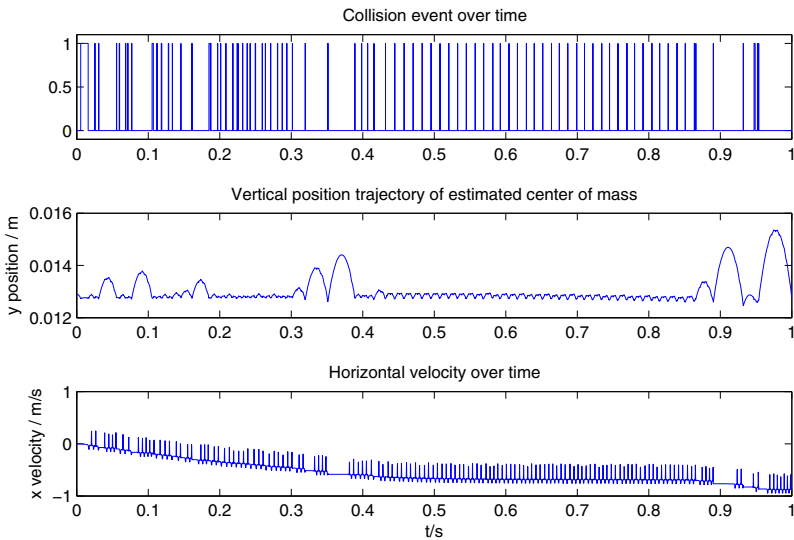


Fig. 8. Modelica simulation results for the millirobot. The top plot depicts the ground collision events over time of all six legs. The second plot shows the relative y position of the motor to the ground. When the simulated robot flips, unstable behavior will be shown in this plot. For this particular setup, it is stable and close to the stable behavior in the experiment. The last plot on the bottom depicts the horizontal velocity.

The motor block, as shown in Fig. 9, is the feedback control of the robot, i.e. the cyber component. Due to the physical properties of the electrical motor, the output torque is affine in the rotational speed input of the motor. This model is directly obtained using the physical properties of the electrical motor used in the prototype.

This motor can be controlled using Pulse Width Modulation (PWM), so that the power is reduced and stability is improved. The PWM will also reduce the torque to prevent joints from breaking. Other improvements are possible. For example, in [27], the authors propose that future models can use additional weight to create a complete 2D planar motion instead of only back and forward motion. The controller for the weight will be more complicated and will require model-based design instead of experimental only methods. Therefore, if more features are to be added to the original prototype, the cyber component will be more complicated and will need to be modified accordingly.

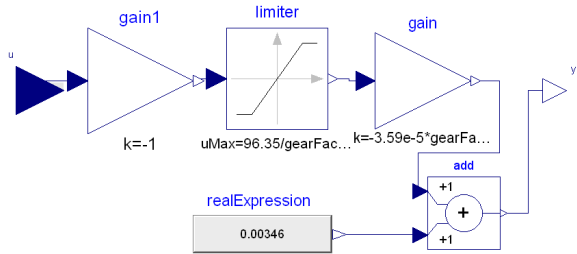


Fig. 9. Modelica motor model. Because of physical nature of electrical motors, the output torque is affine to rotational speed. This model described the particular motor used in the prototype design using its datasheet.

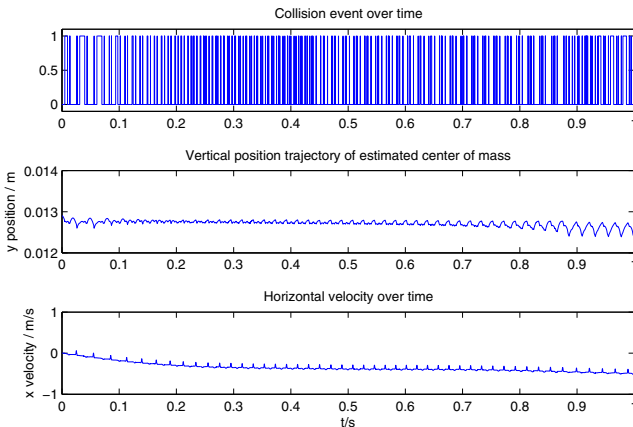


Fig. 10. The stability is improved if the torque is controlled using the sensor input related to the ground contact.

From the second plot in Fig. 8, we note that the robot often bounces away from the ground. The friction force, which is used to move the robot forward, is not present, so such design is not efficient. More importantly, this causes instability in the long term simulation. The subject robot has similar jumping instability

in real experiments, but no solutions have been proposed to improve stability [27]. Now consider a very simple modification of the model that has a PWM motor control unit included so that the power output of the motor is reduced, when the legs of the robot are not in contact with the ground. This makes the model more stable as shown in Fig. 10. To understand further how the PWM motor control helps the stability, one can observe the changes in the associated limit cycle. Fig. 11(a) gives the initial trajectory of θ_l , $\dot{\theta}_l$, θ_r and $\dot{\theta}_r$ within the robot model, where θ_l and θ_r are the generalized coordinates θ_1 of the left and right legs, as shown in Fig. 5. From the trajectory, one deduces the limit cycle of the hybrid system, and reset points due to collision as shown. After adding the PWM motor control, the trajectory takes similar shapes (Fig. 11(b)). The major change is that the swinging speed decreases by about 67%. The converging speed from the reset point towards the limit cycle is faster as well.

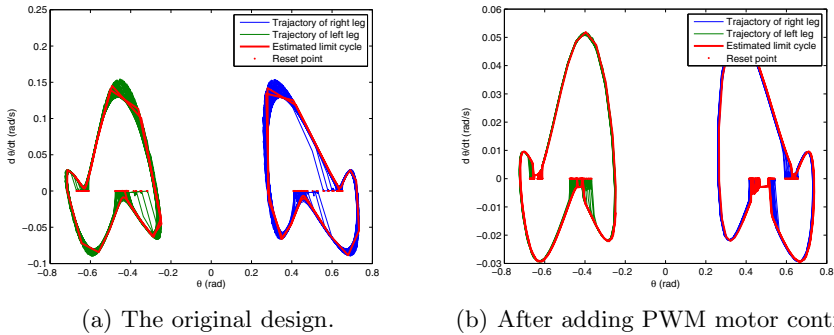


Fig. 11. Fig. (a) (b) describe the trajectories of θ_l , $\dot{\theta}_l$, θ_r and $\dot{\theta}_r$ before and after adding the PWM motor control unit. Compared to the original design, the limit cycle with PWM control takes similar shape, but swinging speeds, $\dot{\theta}_r$ and $\dot{\theta}_l$, decrease by about 67%.

Simulation and system modeling lead to a new design that improves the efficiency of the cyber side of the CPS. The new design also induces the cyber and physical layers to cooperatively behave in a more stable manner.

5 Material Choice and Geometry Exploration

The material selection of the joints is very limited in [27], but in [30] the authors proposed a way of constructing microstructures so that the overall performance of the structure reflects the properties of different material layers. Though this method was mainly used and implemented for thermal bending purposes, this approach can be used in other areas. For microrobots, this means that material selection can consider a much wider range. In the design process, one can design a joint or segments with materials that are unknown but have properties within some reasonable range. In the last step, one can design the microstructure so as to fit the desired properties (specifications).

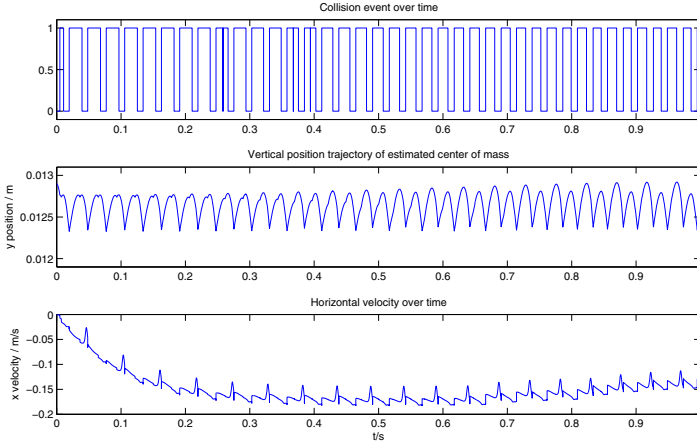


Fig. 12. The stability is improved by reducing the spring constant

In the subject robot [27], the key design parameter is the joint spring constant, which is affected linearly by the elasticity modulus of the material, and is proportional to h^3 . Therefore, the internal torque between the joints can be made 8 times larger by doubling the joint width. Initially the spring constant is chosen so that the internal torque between joints has about the same magnitude as the maximum motor torque. This may induce instability. Fig. 12 shows the result for the same shape of robot structure but with half the joint width, which is still within the reasonable range of joint width in [27]. This design change seems to make the robot more stable. Further

exploration using limit cycle methods gives different results. As shown in Fig. 13, the trajectory shows that the swinging speed decreases by 90%, and the shape of the limit cycle changes. However the limit cycle may be unstable since the trajectory keeps on shifting to the right with no sign of converging.

The material choice can increase the range of possible values for the joint spring constant and even make the joint sustainable under large tension when required in the design. However, changes in the material and geometry of the joints add constraints to the controller, and in particular to the maximum torque output of the motor.

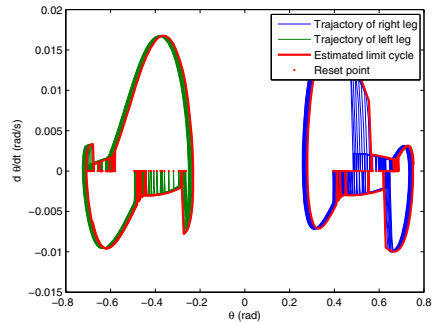


Fig. 13. The trajectories of θ_l , $\dot{\theta}_l$, θ_r and $\dot{\theta}_r$ after using modified joint width. Comparing to the original design, the limit cycle takes different shape, and swinging speed decrease by about 90%.

Now suppose we modify the geometry significantly and the new model takes the shape of Fig. 4b. The shape of the legs is modified to emulate the legs of a crab. The new design is obtained through trial and error to achieve a more regulated walking behavior, i.e. bouncing forward but with similar height.

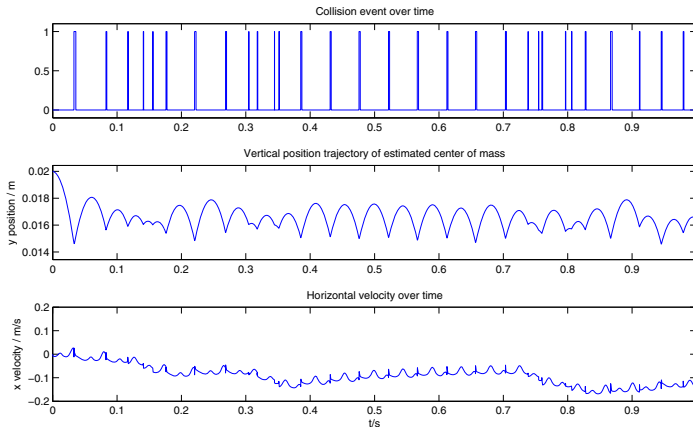


Fig. 14. Collision and motion behavior are different due to different geometry

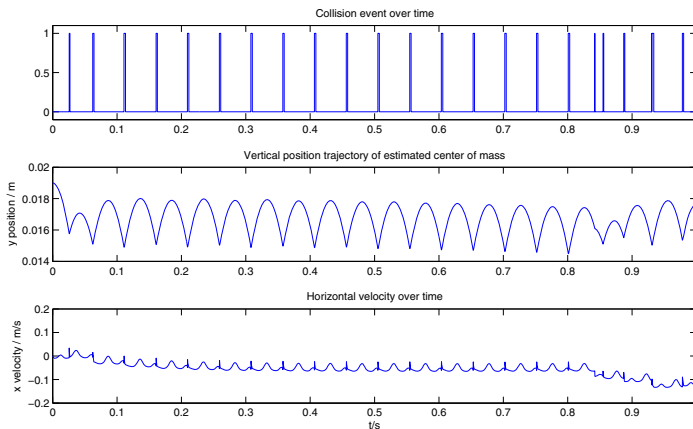
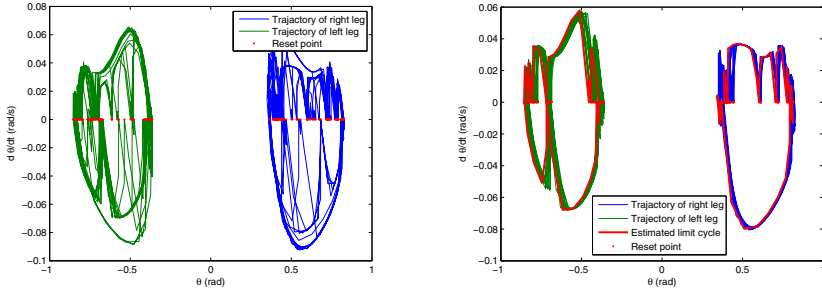


Fig. 15. By adding motor control, as can be seen in the second plot, the jumping behavior is regulated and more stable compared to Fig. 14. The magnitude of the speed is hardly increasing because the robot is constantly lifting off the ground.

As shown in Fig. 14, the robot bounces frequently but it is more regulated and more stable compared to the previous design. Although the joint spring constant has about the same magnitude as in Fig. 12, the behavior of the robot is different. If explored further using the limit cycle, one concludes that the collision for this design actually happened more irregularly, and the limit cycle



(a) Microrobot with modified geometry. (b) After adding PWM motor control.

Fig. 16. Fig. (a) (b) describe the trajectories of θ_l , $\dot{\theta}_l$, θ_r and $\dot{\theta}_r$ of the modified microrobot before and after adding the PWM motor control unit. Compared to the original design, the limit cycle takes different shape. For (a), the collision happened so irregularly that the limit cycle is hardly visible. In (b), the collision is more regular and converges to the limit cycle faster.

is hardly seen as shown in Fig. 16(a). One can use similar control to reduce the instability as discussed in the previous section. The result is shown in Fig. 15 with its associated limit cycle analysis in Fig. 16(b). The jumping behavior is regulated to be more stable but it cannot be removed completely in the case of the new leg shape, which also shows the close relationship between the cyber and the physical components of the robot. Thus the cyber components have to be completely reconsidered as a result of changes in the physical part.

6 Tradeoff and Model Based System Design

As discussed in the previous section, the cyber components have to be adjusted or even redesigned because of the changes in the physical components. A systematic way of jointly considering physical modeling and control design is required. We propose the framework described in Sections 1.2, 1.3, 1.4 for designing millirobots as CPS (Fig. 3). SysML is used as a language for the structure description of the robot, and also used as a linkage with trade-off tools so that material properties trade-off can be performed based on efficiency and stability matrices.

So far the system level design is done in ModelicaML, which is a Java and XML based metamodel which bidirectionally transforms between the UML model and Modelica [31]. As can be seen in Fig. 17, the designer of the robots can use ModelicaML tool (in Eclipse) to modify and simulate design using class definition diagram with underlying simulation engine OpenModelica. One can design control algorithms and have control parameters tuned together with material properties selection and associated trade-offs using the framework described earlier. The model based approach makes the control algorithm easily modifiable, so that different controller designs can be tested and verified.

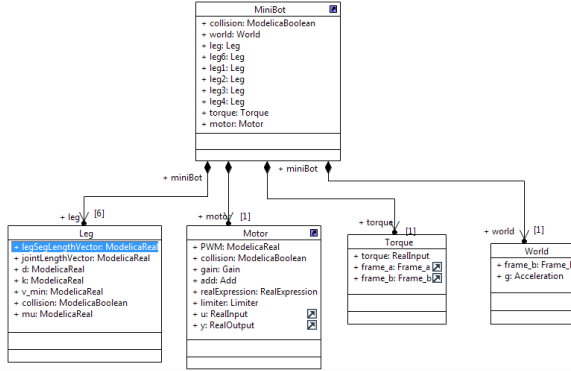


Fig. 17. This is the class diagram created in ModelicaML implementation in Eclipse [31]. The structure model is abstracted from the synchronized Modelica model. The model based approach in developing the Modelica model gives the designer of the robots convenient ways to modify the key physical model parameters (joint width and geometry specified by leg segments length) and the cyber components parameters like PWM. The underline OpenModelica compiler is able to simulate the model at the same time to perform verification tasks for material constraints.

Finite Element Analysis tools like COMSOL for material oriented simulation can be integrated with Modelica to provide a more detailed model of the robot. In this paper a joint is modeled as a torsion spring but clearly it is a simplification of the linkage. One can use COMSOL to provide a more detailed nonlinear model of the joint. The Modelica library FlexBody [32] can be used to solve this problem as well. This library uses the output from FEA tools, such as Nastran, Genesis, to reduce complex finite element models to models which consist of only few boundary nodes, or attachment points.

For this particular prototype, the PWM modification comes from the fact that decreasing torque applied by the motor directly increases the stability. In general this insight can be drawn from the trade-off tools directly. Given material constraints, in particular the deformation condition of the joints, one can formulate this problem as an optimization problem with objective to maximize efficiency and stability matrices such as maximizing forward moving speed and minimizing jumping heights. The tool we implemented in MagicDraw-SysML as shown in Sections 1.2, 1.3, 1.4 can then be used to direct such modifications in the original design. Geometrical modeling and design exploration need to take a different approach from our view. Few modifications of the geometry can dramatically alter the problem. We suggest the combination of design and optimization in earlier stages so that the overall geometry is fairly fixed with only minor changes. This requires the CAD design to be integrated into the design as well. CATIA [33], the 3D CAD tool which can interface with Dymola simulation, might be the best approach for this. We suggest the following design process steps.

1. The designer of the robots defines system structure and high level system design according to the requirement in SysML (System level design) and detailed structure model in CATIA (geometric modeling), alternatively the user can generate SysML/UML from Modelica/CATIA modeling using SysML4Modelica transformation [34] or ModelicaML [31]. The tool will generate model structure as well as Modelica code through model transformation.
2. To do design space exploration such as material trade-off, designer can use FEA tool to generate joint model for several materials so that it can be linked with Modelica through the FlexBody Modelica library. Then the designer has to specify objectives in terms of stability matrices such as jumping height or limit cycle criteria, and performance matrices such as forward motion speed and energy transformation ratio. Constraints, such as the maximum torques the link can sustain, also need to be specified. The trade-off is then done using Consol-Optcad with Modelica simulation.
3. Based on the Consol-Optcad suggestions, the designer has to modify the initial design and go back to system level design and verify that all the requirements are met. If not, the constraints of the previous step have to be refined and the designer will go through the process again until all the requirements are satisfied.

7 Conclusions

To conclude, microrobots are complex CPS, and their cyber part cannot be designed separately from the physical part. In this paper, we described a model-based systems engineering methodology and framework for the design of microrobots as CPS. The physical model and associated control of a particular prototype were examined to demonstrate the close relationship between the physical and cyber parts. We also proposed improvements of the control laws so that the system is more stable and demonstrated these improvements by modeling and simulation. The control laws for the design of this particular type of robot, such as those shown in Fig. 4a or 4b, can be designed via tradeoff with material properties as tunable parameters. They may also need to be redesigned when the geometrical shape changes are significant.

Acknowledgements. We would like to thanks Dana E. Vogtmann for providing experimental demonstrations so we could compare with the models we built. We would also like to thank members of the OpenModelica Association who have kindly helped us solve Modelica related problems.

Research supported in part by the National Science Foundation (NSF) under grant award CNS-1035655 and by the National Institute of Standards and Technology (NIST) grant award 70NANB11H148

References

1. Austin, M.A., Baras, J.S., Kositsyna, N.I.: Combined Research and Curriculum Development in Information-Centric Systems Engineering. In: Proc. of the 12th Annual Intern. Council on Systems Engineering (INCOSE) Symposium (2002)
2. Yang, S., Baras, J.S.: Factor Join Trees in Systems Exploration. In: Proceedings of the 23rd International Conference on Software and Systems Engineering and their Applications (ICSSEA 2011), Paris, France (2011)
3. Wang, B., Baras, J.S.: Integrated Modeling and Simulation Framework for Wireless Sensor Networks. In: Proceedings of the 21st IEEE International Conference on Collaboration Technologies and Infrastructures (WETICE 2012- CoMetS track), Toulouse, France, pp. 268–273 (2012)
4. Yang, S., Zhou, Y., Baras, J.S.: Compositional Analysis of Dynamic Bayesian Networks and Applications to Complex Dynamic System Decomposition. In: Proc. of the Conf. on Systems Engineering Research, CSER 2013 (2013)
5. Yang, S., Wang, B., Baras, J.S.: Interactive Tree Decomposition Tool for Reducing System Analysis Complexity. In: Proc. of the Conf. on Systems Engineering Research, CSER 2013 (2013)
6. Spyropoulos, D., Baras, J.S.: Extending Design Capabilities of SysML with Trade-off Analysis: Electrical Microgrid Case Study. In: Proc. of the Conference on Systems Engineering Research, CSER 2013 (2013)
7. International Council on Systems Engineering (INCOSE): Systems Engineering Vision 2020. Version 2.03, TP-2004-004-02 (2007)
8. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML. The MK/OMG Press (2009)
9. Baras, J.S.: Lecture Notes for MSSE class, ENSE 621 (2002)
10. Haskins, C., Forsberg, K., Krueger, M., Walden, D., Hamelin, D.: Systems Engineering Handbook. INCOSE, San Diego (2011)
11. The eMofflon team: An Introduction to Metamodelling and Graph Transformations with eMofflon, V 1.4. TU Darmsadt (2011)
12. Anjorin, A., Lauder, M., Patzina, S., Schurr, A.: eMofflon: Leveraging EMF and Professional CASE Tools. In: INFORMATIK 2011, Bonn (2011)
13. No Magic, Inc.: Open API-User Guide. Version 17.0.1 (2011)
14. Meyer, J., Ball, M., Baras, J.S., Chowdhury, A., Lin, E., Nau, D., Rajamani, R., Trichur, V.: Process Planning in Microwave Module Production. In: Proc. SIGMAN: AI and Manufacturing: State of the Art and State of Practice (1998)
15. Fan, M.K.H., Tits, A.L., Zhou, J., Wang, L.-S., Koninckx, J.: CONSOLE-User's Manual. Technical report, Un. of Maryland, Vers. 1.1 (1990)
16. Fan, M.K.H., Wang, L.-S., Koninckx, J., Tits, A.L.: Software Package for Optimization-Based Design with User-Supplied Simulators. IEEE Control Systems Magazine 9(1), 66–71 (1989)
17. Tischler, M.B., Colbourne, J.D., Morel, M.R., Biezad, D.J.: A Multidisciplinary Flight Control Development Environment and its Application to a Helicopter. IEEE Control Systems Magazine 19(4), 22–33 (1999)
18. Potter, P.J.: Parametrically Optimal Control for the UH-60A (Black Hawk) Rotorcraft in Forward Flight. MS Thesis, Un. of Maryland (1995)
19. Nagy, Z., Ergeneman, O., Abbott, J., Hutter, M., Hirt, A., Nelson, B.: Modeling assembled-mems microrobots for wireless magnetic control. In: Proc. of IEEE Intern. Conf. on Robotics and Automation, ICRA 2008, pp. 874–879. IEEE (2008)

20. Mohebbi, M.H., Terry, M.L., Böhringer, K.F., Kovacs, G.T.A., Suh, J.W.: Omnidirectional walking microrobot realized by thermal microactuator arrays. In: Proc. of 2001 ASME Intern. Mech. Engin. Congress and Exposition, pp. 1–7 (2001)
21. Donald, B., Levey, C., McGray, C., Paprotny, I., Rus, D.: An untethered, electrostatic, globally controllable mems micro-robot. *Journal of Microelectromechanical Systems* 15, 1–15 (2006)
22. Bergbreiter, S., Pister, K.: Cotsbots: an off-the-shelf platform for distributed robotics. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 2, pp. 1632–1637. IEEE (2003)
23. Erdem, E., Chen, Y.M., Mohebbi, M., Suh, J., Kovacs, G., Darling, R., Öandhringer, K.B.: Thermally actuated omnidirectional walking microrobot. *Journal of Microelectromechanical Systems* 19, 433–442 (2010)
24. Churaman, W., Currano, L., Morris, C., Rajkowski, J., Bergbreiter, S.: The first launch of an autonomous thrust-driven microrobot using nanoporous energetic silicon. *Journal of Microelectromechanical Systems* 21, 198–205 (2012)
25. Hiller, J., Lipson, H.: Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics* 28, 457–466 (2012)
26. Fritzson, P.: Introduction to Modeling and Simulation of Technical and Physical Systems. Wiley-IEEE Press (2011)
27. Vogtmann, D.E., Gupta, S.K., Bergbreiter, S.: Multi-material compliant mechanisms for mobile millirobots. In: Proceedings 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 3169–3174 (2011)
28. Bellouard, Y.: Microrobotics: Methods and Applications. CRC Press (2010)
29. Otter, M., Elmqvist, H., Dí López, J.: Collision Handling for the Modelica Multi-Body Library. In: Proc. of the 4th Intern. Modelica Conf., pp. 45–53 (2005)
30. Shin, M., Gerratt, A.P., Metallo, C., Brindle, A., Kierstead, B.P., White, R.D.: Characterization of a micromachined parylene-based thermal c-shape actuator. *Journal of Micromechanics and Microengineering* 21(9), 095028 (2011)
31. Schamai, W.: Modelica Modeling Language (ModelicaML): A UML Profile for Modelica. Tech. Rep. 2009:5, Linköping University, Department of Computer and Information Science (2009)
32. Dymola libraries flexbody | claytex,
<http://www.claytex.com/products/dymola/model-libraries/flexbody-library>
33. CATIA systems engineering,
<http://www.3ds.com/products/catia/solutions/catia-systems-engineering>
34. SysML-Modelica Transformation (SyM), <http://www.omg.org/spec/SyM/1.0>