Response to Reviewer 4 Comments

For research article applsci-3447801

1. Summary

Thank you very much for taking the time to review this manuscript. Please find the detailed responses below and the corresponding revisions denoted by red color in the re-submitted manuscript. Reviewer's suggestions are provided readable and improved text.

2. Questions for General Evaluation	**Reviewer's Evaluation**	**Response and Revisions**
Does the introduction provide sufficient background and include all relevant references?	Can be improved	*Answered below*
Is the research design appropriate?	Yes	
Are the methods adequately described?	Can be improved	*Answered in Point 3, Responses 2-4.*
Are the results clearly presented?	Yes	
Are the conclusions supported by the results?	Can be Improved	*Answered in Point 3, Response 3.*

Thank you for pointing this out. We agree with this comment. Therefore, we have expanded the existing Introduction with an additional research review from line 58, page 2 to line 100, page 3, and we have added one sentence from line 116 to line 118, page 3. The texts are given below.

- line 58, page 2 to line 100, page 3

"[The popularity of XML-based languages has increased over the past years due to the possibility of exchanging data between heterogeneous systems, and numerous studies discuss them: XML-based Robot Description Language for describing robot structures was presented in [16], RoboML based on XLM for robot programming in [17]; AVSML (Agent and Virtual Space Markup Language) an XML-based markup language for integrating web information into virtual 3D space in [18].

In the work of [19], an application for the robot palletizing process is developed in the ABB native language RAPID with an interface written in XML has ability to work together with the software for offline robot programming RobotStudio.

An XML is used as the basis for a new language to specify video game - Video Game Description Language (XVGDL) in [20].

One of the most common robot description formats - Unified Robot Description Format (URDF) was introduced in 2009 [21,22], with the development of an open source set of software libraries and tools for building of robot applications, named the Robot Operation System (ROS). URDF is based on XML technology, and was applied to describe the kinematic structures of robots, their dynamics and geometry [22,23]. The self-calibration procedure of a low-cost 5-axis 3D printer, where the calibration data was used to generate a kinematic model in URDF format, was proposed in [24]. Procedures for

automatic generation of URDF files from Solid Works are the subject of research in [25,26]: for the visualization of a modular robot manipulator in [25] and a micro-assembly robot in [26].

Similarly to URDF, the XML-based Simulation Description Format (SDF) has emerged, expanding the possibilities of developing a robotic environment [23]. Because both URDF and SDF are XML-based, they follow the rules of XML syntax, making them both human-readable and machine-parsable. SDF was used to describe Building Infor-mation Modelling objects and integrated with the finite element method (FEM) and ROS to assess structural stability in [27].

An overview of robot description formats and methodologies for visualizing robots and their environment is presented in [28].

Visualization of systems and devices, i.e., their virtual 3D models and digital twins (DT), are recognized as a potential solution for the improvement of these processes and systems and therefore attract extreme attention from the academic community and industry. An XML is one of the most commonly used technologies to develop the digital twin. In the paper [29], due to the analysis of the sheet metal punching process, different configurations of sheet metal punching machines were developed with a complete virtual 3D scene in the form of XML files. On the example of a digital twin of the overhead crane, in [30] it is presented a system framework for its integration into an Extended Reality (XR). The application of digital twins as possible answers on challenges in production systems is the subject of research in [31,32]. In [31], an overview of the possibility of Digital Twins to support Human-Robot Collaboration in production systems is presented as well as a comparison of key commercial software regarding the possibility of fulfilling the requirements for DT development. The review of research in [32] pointed out numerous gaps in the application of DT in production systems as well as indicated directions for future research.*]"*

- line 116 to line 118, page 3

*"[*A comparative study of Gazebo and Unity 3D in the development of a virtual model of the collaborative robot UR3 and its application for assembly work in production is presented in [53].*]"*

3. Point-by-point response to Comments and Suggestions for Authors

The article can be published after some explanations

Comments:

This paper proposes novel approach to 3D virtual modeling of articulated mechanisms. It follows widespread use of XML for various applications and defines a version of XML specially designed for description of 3D geometric models of articulated bodies. It was shown how the 3D geometric model of a mechanism can be gradually developed using suitable defined elements and stored in corresponding XML file. The developed XML model is processed and using powerful VTK (Visualization Toolkit) library the corresponding virtual model is built and shown on the computer screen. To drive the virtual model the dynamical model of the mechanism is developed using Bond Graph modelling techniques. The models, 3D virtual geometric and dynamic, are created using the corresponding software packages: BonSim3D Visual and BondSim. The models are interconnected by a two-way named pipe. During the simulation of dynamical model, the parameters necessary to drive the virtual model (e.g., the joint displacements) are collected and sent to the virtual model over the pipe. When the virtual model receives a package the computer screen is updated showing the new state of the mechanism. The approach was demonstrated on example of a holonomic omnidirectional mobile robot.

- The article presents a unique combination of XML-based geometric model descriptions and Bond Graph methods for creating dynamic mechanism models. This approach enables the development of accurate digital twins of mechanisms that can interact in real time.
- Integration of virtual and dynamic modeling – XML is used to create 3D models, while Bond Graph is applied for dynamic analysis, ensuring an efficient two-way connection between the models. The described methodology can be applied not only to robots but also to other complex mechanisms such as manipulators, multi-legged platforms, cranes, and so on.
- Practical value – the approach can be applied not only to robots but also to other complex mechanisms, such as manipulators and cranes.
- Detailed explanation – the article provides a step-by-step methodology for building models, including calculations, diagrams, and algorithms.
- Experimental validation – the simulation results are confirmed by comparing data from the virtual and dynamic models, demonstrating their accuracy.

Negative aspects:

Comment 1:

Implementation complexity - the proposed method requires knowledge in XML, Bond Graph, and visualization libraries (VTK), which may complicate its use for unprepared users.

Response 1:

Thank you for pointing this out. We agree with this comment. Therefore, we have written our answer below.

The knowledges in XML, VTK and Bond Graphs are welcome, but BondSim3DVisual and BondSim programs hide many aspects of them. Thus, regarding XML, the user really does not meet the real XML, but is dealing with construction of a DOM (Document Object Model) like tree. The software helps him in doing this. What is important is that the user understand the design of the mechanism, i.e., the body parts that it consists of, how they are connected, type of the joints and the movements they allow, etc. He or she gradually build the model tree starting from the base and going up. The tree structure practically has a standard structure as shown in Figure 10. If he or she want to see underlying XML form, it is already here; it is automatically generated while building DOM like tree. It is written on read-only XML Output document, and we may see it simply by clicking the document tab (Figure 3.).

Regarding the VTK, the user does not need to know about it, even that it exists. It is hidden from the user by the program and is implemented inside it. The generating the 3D bodies proceed completely automatically while processing the model DOM like tree.

Now what to say about Bond Graphs. There are many books written on this topic. Even the first author V.D. wrote two of them (that was published by Springer Verlag, Heidelberg, Germany).

The theory that we propose is relatively simple. We added it as Sect. 3.1. Introduction to Bond Graphs (pp. 12-16, lines 397 - 552). We build bond graph model as a tree of the components. There are word components that represent the branch of the model tree, and the models of elementary processes represented by nine elementary components as model tree's leaves. To develop the model, we follow the corresponding physical theory, in this case the Mechanics.

To generate the model of a mechanism we apply system decomposition. At the start we decompose the subject of the problem on the System and its Environment. Then, analyze the System, identify the basic components and how they act one on the other and with environment. We model the components by the word (textual) model having the ports by which they interact and to indicate it we connect them by the corresponding lines – bonds. The ports in Bond Graphs are in the forms of half-arrows indicating direction of the power flows through the system. Thus, we obtain the basic level bond graph model.

We continue with decomposition of every component on its parts and construct the corresponding Bond Graph model documents. We continue until we reach the basic physical processes, which are modeled by Bond Graph elementary components. Their ports are characterized by a pair of variables – effort and flow. The effort may represent force, moment of force, electrical voltage, etc.; similarly, the flow can be velocity, angular velocity, electrical current, etc. The relation between them is the constitutive relation of the element stored in the port. The inertial and capacitive elements have in addition the state variables of the elements – the generalized momentum and charge.

The BondSim help the user in developing the model, constructing its tree structure, and generating the mathematical model from it. The model is solved by BDF (Backward Derivative Formula) type solver developed specially for this purpose. There is only one solver, which is capable of solving models of the indexes up to 2. I dare to say it is better than in other simulators.

Comments 2:

Limited demonstration - although the article focuses on omnidirectional mobile robots, its universality for other mechanisms is not confirmed by additional experiments.

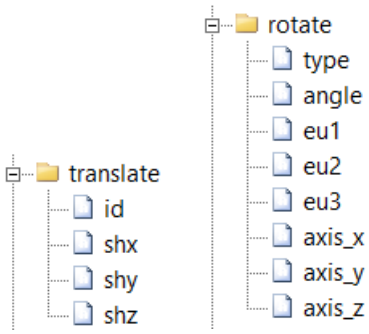
Response 2:

We agree. We have, accordingly, added the explanations to this point.

There was not enough place for treatment of the other mechanisms because of the length of the paper. However, XML structure in Figure 10 can be applied to other mechanism as well. In addition, we add Figure 7, which gives the details of transform elements used, and lines 281-291, which specify the drive elements. Thus, all joints of the practical interest are covered. (See please below.)

XML define elementary geometric element from the ordinary point, cuboid, circular and general closed cylinder with holes, the bodies of revolution, 'stl' files CAD parts, and other. Practically, the BondSim3DVisual software is a shell for a part of VTK used for generating rigid bodies. XML is simply its input language. It has also important property that it supports generating named part server.

*"[*Often transforms along the body links are much simpler. Thus, e.g., joint coordinate frame fixed to the previous link can be used simultaneously as the link frame. In that case, pre- and/or post-transforms are not necessary (they are identity transforms); only the drive transform is needed.



Type	Kind	Rotation angles
x, y, z	elementary	angle
zxz, zyz	proper Euler	eu1, eu2, eu3
zyx, xyz	Tait-Bryan	eu1, eu2, eu3
axis	finite rotations	angle axis_x,axis_y,axis_z

(a) **(b)**

Figure 7. (**a**) Translate element; (**b**) Rotate element with the corresponding attributes (on the right)

The <drive /> element represent link diver, which can be real or virtual. In any case it is defined by *type* attribute, which can be one of the following diver types: prismatic_z, prismatic_y, prismatic_z, revolute_x, revolute_y, revolute_z, and spherical. First six types correspond to single degree of freedom joints, and the last is three degree of freedom rotation about an axis through center of the joint.
]"

We already have published several papers that are dealing with different mechanism:

1. *Four bar mechanism, ABB IRB 4400 robot,*

   Damic, V[jekoslav] & Cohodar, M[aida] (2016). Dynamic Analysis and Visualization of Spatial Manipulators with Closed Structure, Proceedings of the 26th DAAAM International Symposium, pp.0109-0117, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-07-5, ISSN 1726-9679, Vienna, Austria DOI: 10.2507/26th.daaam.proceedings.016.

2. PUMA 560 robot arm,

   Damic, V., Cohodar, M., Damic, D. (2014). Multibody Systems Dynamical Modeling and Visualization based on IPC technique, the proceedings of the 2014 International Conference on Bond Graph Modeling and Simulation – ICBGM'2014, Edited by Jose J. Granda, Dean C Karnopp, Simulation Series, Vol.46, No.8, pp. 773-778, ISBN: 978-1-63266-700-7, The Society of Modeling & Simulation, Monterey, USA, July 6-10, 2014.

3. Scara type robot ABB IRB 910 SC 450

   Damic, V, Cohodar, M. (2016). Dynamic analysis of robot manipulators: a component model approach, Proceedings of International Conference on Innovative Technologies, IN-TECH 2016, Prague, 06. - 08.09.2016.

4. Spherical manipulator arm

   Damic, V., Cohodar, M. (2014). Procedure for Visualization and Dynamic Analysis of Robot Manipulator, *Journal of Trends in the Development of Machinery and Associated Technology*, Vol.18, No.1, ISSN 2303-4009 (online), pp. 127-130, (EBSCO)

5. Slider-crank mechanism

Damic, V., Cohodar, M. (2015). Multibody systems with closed structure: Dynamic analysis and visualization of slider crank mechanism, *Journal of Trends in the Development of Machinery and Associated Technology*, Vol.19, No.1, ISSN 2303-4009 (online), pp. 173-176

Comments 3:

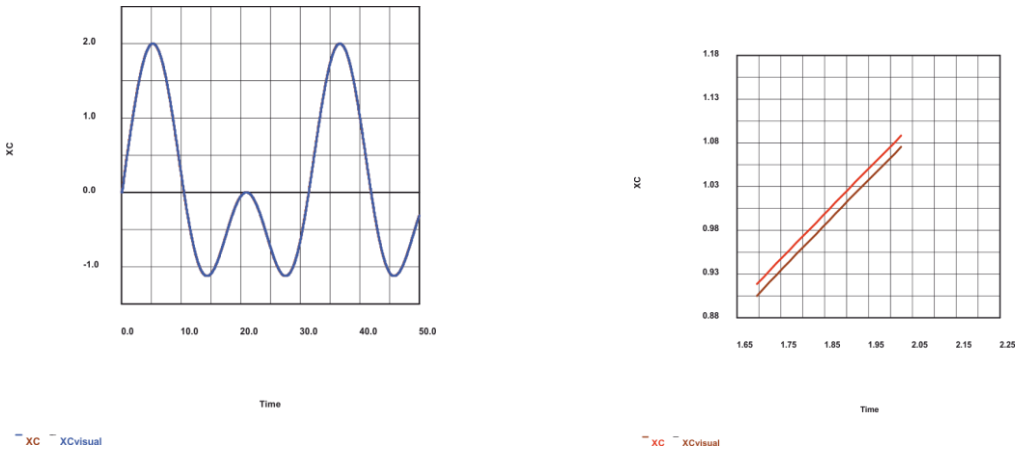
Lack of performance analysis - the article does not address potential performance limitations or delays when operating two interacting models in real-time.

Response 3:

We agree with your kind comment and upgraded the paper with explanation of simulation results shown below (Sec. 4, pp. 30-31, lines 909-916, Figure 33a and b).

"[

To further explain the obtained results, we put x-coordinates of the center point XC and XCvirtual on the same y-axis of the display at the right of IPC component. It is shown in Figure 33a. At first glance they seem as a single curve. We expand a small rectangle around a point on the curve. The generated enlarged plot is shown in Figure 33b. There are two curves displaced by 30 ms (milliseconds). Thus, signal sent back over the pipe is delayed by this time value, and as the result the accuracy of this signal returned is of moderate accuracy as seen from Figure 26b. The delay depends on the frequency of sending data packs over the pipe. In this simulation it is exactly the same number, i.e., 30 ms.



(a) (b)

Fig. 33. Delay of signal transport: (a) The both signals displayed on the same axis, (b) Time delay between the signals.

]"

Comments 4:

Potential scalability issues - the approach may be challenging to implement in larger systems with multiple interacting components.

Response 4:

We thank you for the comment. We give the following observations. Consider first the 3D virtual side. If we have highly populated scene, with complex equipment (robots, humanoids, transport system, etc.) we may develop the XML without much problems. The XML input language support the systematic decomposition of the scene and modeling at each level of decomposition. In Robotino problem there are five levels: project, robot (the Robotino), and three objects (the wheels). The separate XML models are short and there is really no problem with them.

The visualization of the scene is responsibility of VTK. The models we deal with are relatively simple for VTK and they can be executed easily. It is, however, necessary to have enough computer resources. Note that VTK is applied to many software systems from engineering, medicine, business and other. It is really extremely powerful library.

The BondSim is a baby of V.D., so he is responsible for its performances. It is based on Bond Graph theory in the form formulated by Prof. Paynter, and Karnopp et al [85], but developed further. First it introduced the concept of general component, which from outside looks as common Bond Graph word (textual) model, but it is associated with a document that is used for definition of its internal model. To support modeling a BondSim classes are defined, which completely describe structure of the Bond Graph model (components, documents, ports, bonds, etc.). It is written in Microsoft Visual Studio C++/Cli.

To store a model the Microsofts' structured storage files are used, i.e., a file system inside a file. They contain storages and steams; the first play role of common directories, and the other of the files. Thus, every Bond Graph component in the model tree is represent by a branch, and is stored into the project file as a storage. On the other hand, the component models are the storage's streams. Thus, when we open a component and edit its Bond Graph models, when closing its document, we have several options:

- reject storing
- local storing into the component document object in the memory
- storing the complete tree to the project object

When a modeling project is closed it is stored in the file by visiting all component documents, generating (if they do not exist) the corresponding storages and storing in the corresponding streams. The model file from outside looks as a common file, and has the extension "bnd" (i.e., the bond) and is generally pretty large. Thus, e.g., the Robotino file has size of 8.69 MB. Its name was defined internally by the program. We access it by user friendly name, e.g., "Robotino".

The mathematic equation that models corresponding element has very simple form consisting of few terms. The number of equations corresponds to the number of elementary components. The BondSim program translate is from the common form used (call infix form, i.e. operand operator operand, e.g. a+b) into prefix form (e.g., +ab) for easy processing and evaluation. The Robertino project studied in this paper has 244 equations. Its partial derivate matrix has 555 elements; thus, it is very sparse (2 to 3 elements per equation on the average). When processing the mathematical model, the elements of partial derivative matrix are evaluated by symbolic partial differentiation.

The BondSim variable coefficient form of BDF algorithm, of maximal order six, to solve the mathematical model. It is the most stable version of the code, and can solve semi-explicit DAEs

(Differential Algebraic Equations) up to index 2. The solver asks for frequent evaluation of the partial derivative matrix evaluation. Due to is very sparse form, the analytic expressions for its element, and use of the .Net assembly, it can be solved very efficiently. Thus, e.g., the simulation of the Robotino model corresponding the simulation time = 50 s, maximum step size = 0.015 s, maximum output interval = 0.015 s, the relative and absolute error tolerances of 10^{-6} is 0.36 s when it is not connected to the 3D virtual model. When, it is connected to it, and time delay for the sending the data pack is set at about 30ms (30 packs per second), and video recording was switched on, it is much larger, i.e., it is about 80 s.
Every effort was made to enable solving the large models. The program is 64bit version, and thus it is restricted only by the size of the computer memories. The largest model that was encountered so far was about 2,500 equations, i.e., an order of magnitude larger than Robotino project.

Comment 5:

Absence of comparison with alternatives - the article lacks comparative analysis of the method's advantages relative to other approaches to modeling and visualizing mechanisms

Response 5:

Our approach to modeling is based on the geometrical and dynamical modeling of working space containing the articulated mechanisms and the other objects. The geometrical modeling result in the model that closely resembles the structure and look of the equipment we are dealing with. To achieve this, we systematical build the model by XML elements that corresponds to the real building blocks of the mechanism. The model is built in the form of a DOM (Document Object Model) like model tree. By processing the tree and using classes defined in VTK (Visualization Toolkit) we develop a 3D virtual model of the equipment. To simplify geometric modeling of artificial mechanism we developed software BondSim3DVisual, which simplify XML modeling by helping a user building the model tree, and processing it into a 3D virtual model on the computer screen. It is really a shell around a part of VTK library dealing with representation of the bodies and coordinate transformations, and their rendering on the computer screen. It uses an XML as input language. XML is used in robotic literature as text in Point 1 indicates. It is mostly oriented to robotic operations. Our approach was directed to constructing the geometric models that follows the structure of the mechanism

We believe that this approach is unique and very powerful. It is possible to apply it on different mechanisms not on the robots only. We may model a scene consisting of different objects: robots, humanoids, transport systems and others. Every particular object can be modeled separately and stored in the program database. To develop the project scene's model the object models from program database are placed at the scene. There is no restriction on the size of the scene. However, a large scene asks for stronger computer resources to effectively visualize and update the scene. The scene with a Robotino mobile robot was

selected because of its complex mechanical structure. We showed that virtual model of it can be systematically developed.

The paper is updated by the explanations in Sec. 2.2 Generating virtual 3D model, pp. 5-6, lines 210-223) as shown below.

*"[*The approach that we use is general purpose and can be applied to different articulated mechanisms, not only Robotino. The building of the XML model starts at a root and continue by adding different elements and its attributes systematically following the structure of the real equipment. For this purpose, a wide range of XML elements are at disposal ranging from simple geometrical shapes such as cuboid, cylinder, sphere, bodies of revolution, CAD designed parts, and the others, to joints for interconnecting of bodies, and different type of transforms, which are used to transforms XML elements body models between different coordinate systems and driver units. In this way a virtual model is obtained, which has very close the look and structure as the modeled equipment.

There is no restriction on number of the objects that the working space can contains. To that purpose a *project* root is used in which we may insert different items from application data base (XML models of robots, humanoids, transport systems and others). However, as population of the working scene increases a stronger computer resource necessary in order to efficiently update the computer screen.*]"*

The virtual models by its structure are the static geometrical objects. We may drag them, or even manipulate them manually. To force them to move we need to generate the signals and transfer them to its joints' drives (Figure 6b in the paper). These signals may be generated by an external dynamical model of the mechanisms.

To develop the dynamic model of the mechanism (Robotino) we use Bond Graph methodology. It is the fact that differs our approach of others. Thus, e.g., MATLAB/Simulink uses a block diagram environment. For complex problems it uses mathematically based input languages, which are by a suitable coder translated into C or C++ language. A similar approach uses the Dymola simulator. To develop a dynamical model, we use the BondSim program. It is a software application developed many years ago by the first author (V.D.), and described detailly in ref. [62] of the paper. As shown there it can be applied to different problems ranging from the fields of Mechanical Eng., Electrical/Electronic Eng., Control Eng., and the others. Thus, we develop the dynamical model of Robotino using component model approach of BondSim. It is based on Bond Graphs, but with the causality issues (input-output relationships) of the classical Bond Graphs rejected. The models are developed in the form of DAE (Differential Algebraic Equations) as described in Sect. 3.7. of the paper. It seems that Simulink DAE solver is of index 1 (perhaps DASSL, Brenan et al [88]). We believe that similarly hold for the other simulators. Note that classical Lagrange model of rigid body dynamics is of index 3, thus solving robotic model is not an easy problem. Characteristics of Bond Graphs is that it deals with velocities, not the positions, thus the rigid body dynamical models are DAEs of index 2. Thus, it is solvable.

At time we started developing BondSim (eighties of the last century) use of DAEs were rare. Now many software switched to DAE's solvers, usually in the form of popular DASSL code. Our approach to BDF (Backward Differentiation Formula) code is still at advantage:

- We use the variable coefficient form of BDF (DASSL uses the constant coefficient form)

- Evaluation of partial differentiation matrix is done by symbolic differentiation of model equations (DASSL uses numerical differentiation).
- Use of assembly code generated by the application for evaluation of models and the matrices.
- Special algorithm for determining the initial values of the all the system.

The developed model follows the physical structure of the mechanism and thus, it is very close to its physical model, i.e., similarly as its physical origin.

4. Response to Comments on the Quality of English Language

Point 1:

(x) The English is fine and does not require any improvement.

Response 1:

OK

5. Additional clarifications

The comments that were sent to us are really elaborate and helpful. We express our great thanks to the reviewer and editor on large effort and time used**.** Many comments are just concerned with the missing of longer/deeper explanations. While writing the manuscript we as authors struggled between what to include into the manuscript, and to have it to be of a moderate length. Now with all additional explanations we have the manuscript of 36 pages.

We, and in particular V.D., are dealing with Bond Graph modeling for long time. V.D. had developed a software for simulation of dynamical system even in eighties. It was written in Fortran that was the main computer language at that time, particularly in engineering. Later, at beginning of nineties, on advice from a college from UK, switching to C++, really Microsoft Visual C++ based on MFC (Microsoft Foundation Class) library, was made.

There were some changes with respect to the classical Bond Graphs. First, some physically not justified approach was rejected resulting in a simplified Bond Graph theory. It was wanted to separate modeling from the model solving. Thus, the causality issue, inherited form block-diagram approach of control theory, is not necessary and are rejected. In Newton Mechanics, e.g., a force acting on a body and its velocity (or momentum) are neither the input nor the output. His laws do not speak about this. Thus, without the causalities, the mathematical model that Bond Graphs imply are semi-explicit DAE. To solve them the DASSL solver, which can solve the DAE up to index 1, was not used but instead it was developed a separate solver based on general variable coefficient BDF (Backward Differentiation Formula), which is capable of solving the DAE up to index 2 (See Sec. 3.7.).

The BDF needs the partial derivative matrix (a form of a Jacobian) of the model equations. Due to the constitutive relation of Bond Graph elements in the analytic form, a symbolic processing of them are conducted to evaluate mathematical expressions of the matrix elements. Thus, the matrix elements are easily analytically evaluated; there are no need for numerical differentiation as in DASSL).

The BondSim as Bond Graph modeler and simulation application has only one solver, not several of them as the other simulation languages (MATLAB/Simulink, Dymola and others); i.e., the powerful BDF one. Thus, we think BondSim is at least as powerful as others, and due to Bond Graphs, easier to apply to develop the complex physical models. It does not need mathematical knowledge higher than that of high school, but good understanding of the physics of the problem, in this case the Mechanics.

In this paper we select to apply it on Robotino due to its complex mechanical structure. We show that we can systematically develop the physical model to it's the smallest details. If you look at our video on You Tube, you can see that wheels and rollers really rotate; it is not only the body moving over the screen. In the published models solved by the former mentioned simulators we didn't see such complex examples.

The visualization software BondSim3DVisual is newer. The version of the software was described in ref. [45]. It uses C-like language for description of the geometry of the 3D articulated mechanism. It was not easy to use so it was switched to XML and develop it in the present form. Now it is more intuitive and easier to use. In the new version of software, a DOM like XML input was developed using MFC, inspired by Microsoft XML Notepad and File Explorer. Processing was retained as it was in the first version and is based on VTK (Visualization Toolkit). It is completely independent of BondSim. However, to enable IPC communication with other software, it can create a named pipe server following Microsoft implementation (that is available on every Windows PC). Similarly, BondSim can connect server in a similar way.

This rather length explanation we need to give to more clearly describe our motivation.