IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 4, 2017, accepted November 13, 2017, date of publication November 17, 2017,
date of current version December 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2774835

Automatic Generation of Pipelines Into a 3D Industrial Process Model

SEPPO ANTERO SIERLA[1], TOMMI A. KARHELA[2], AND VALERIY VYATKIN[1,3], (Senior Member, IEEE)

[1]Department of Electrical Engineering and Automation, Aalto University, 00076 Aalto, Finland
[2]VTT Technical Research Centre of Finland, 02044 VTT, Finland
[3]Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 971 87 Luleå, Sweden

Corresponding author: Seppo Antero Sierla (seppo.sierla@aalto.fi)

This work was supported by the Engineering Rulez Research Project through Tekes, Technical Research Centre of Finland, Aalto University, Equa, Fennovoima, Fortum, Masinotek, Outotec, Prosys OPC, PSK, Pöyry, and Semantum.

ABSTRACT Simulation has become an established technique to support the design of complex, mechatronic or cyber-physical systems. Ideally, simulations should already be performed at an early design phase before high-cost design commitments are made, and the recent advances in the digitalization of design information open possibilities for automatic generation of simulation models. However, high-fidelity simulation model building depends on accurate data. In particular, first-principles models are desirable source information for simulations, but such models generally are not available at an early design stage. This paper investigates the automatic generation of first-principles 3-D models for piping intensive systems based on design information that is available at an early design stage, namely piping & instrumentation diagrams (P&ID). An algorithm is presented for the generation of such 3-D models based on machine-readable P&ID information. The main focus of the algorithm is the automatic generation of feasible pipelines into the 3-D models, so that the model has sufficient information, which can be exploited in further work to automatically generate high fidelity first-principles thermo-hydraulic simulations.

INDEX TERMS Development lifecycle, first-principle model, model generation, pipelines, simulation.

I. INTRODUCTION

Simulation models of industrial processes can significantly support engineering activities along the lifecycle of an industrial plant. One obstacle to exploiting simulation in the early design phases is that the source information for the models may not be available at that time. A second obstacle is that the development cost for accurate models is a significant factor [45], motivating work in automatic model generation. Despite the obstacles, simulation would be particularly useful in the early development phases of complex, mechatronic or cyber-physical systems (CPS). Simulation can reveal crucial insights on the impact of design choices to important system properties before high cost design commitments are made [32], [63], and can resolve testing related challenges is modern product development processes with increasingly tight schedules [57]. Previous work in this area has addressed such issues as reliability and safety [53], [54], emergent behavior [44] and automatic generation and assessment of simulation models of alternative designs [55].

This paper addresses the automatic generation of industrial process models from piping diagrams. Especially for 3D model creation, the design of pipelines is a work-intensive task for diverse industrial systems such as:

- plants in petrochemical, conventional power, nuclear power, mineral processing, pharmaceutical and food & beverage industries;
- fuel, cooling and HVAC systems in aircraft, vehicles, ships and buildings;
- oil, gas, water and waste water piping in urban infrastructure and critical infrastructure.

At an early design stage for such systems, a P&ID (Piping & Instrumentation Diagram) is often available. This diagram identifies the process components, the pipelines and the instrumentation required by the automation system. However, it does not have any information on the positions of the components or the routes of the pipes, since the route of the pipeline on the 2D diagram cannot be assumed to have any correspondence to its route in the 3D model that

will be developed at a later design stage. In recent works about simulation model generation from a P&ID [5], [8], authors claim to generate low fidelity models best suited for the Factory Acceptance Test phase that comes late in the development lifecycle. Our ambition is to generate models that have sufficient information for a simulation package that has a thermo-hydraulic solver and is thus able to dynamically determine process variables such as temperature, pressure, vapor pressure and chemical concentrations in various normal and abnormal operating scenarios of the plant. In [39] and [54], such simulation models were created from early phase designs, but instead of generating them from a P&ID, the models were built manually and important properties affecting thermo-hydraulic behavior, such as pipe lengths and the height of nodes, were set arbitrarily.

The research goal in this paper is to be able to generate a 3D model of tanks and piping based on the information in a P&ID, which the designer augments with 3D position information of the tanks. Thus the main task towards realizing this goal is to develop an algorithm for generating the pipelines in 3D. The model generation is not fully automatic, but bearing in mind that excessive model development cost is an obstacle for industrial application [45], it is advantageous to make minimal demands on the process designer's time. Thus our goal is to propose a method for model generation that satisfies all of the following requirements:

1. all source information for the model generation should be available early in the design lifecycle;
2. the model generation results should also be ready early in the lifecycle;
3. compared to previous work that satisfies requirement 1 [5], [24], a significantly higher level of model fidelity is desired. In particular, the generated pipeline model should have the information required for a first-principle model as defined in [45];
4. the workload for the process design engineer should be low.

The benefit of generating a visual model is to enable a very rapid method by which an expert process designer can inspect the visual design and make adjustments to tank positions or to force the rerouting of pipelines through certain locations, resulting in an early phase 3D which is feasible according to the judgment of the expert process designer. The main envisioned usage of such a model is that it has sufficient source information from which it will be possible, in further research, to automate the generation of a thermo-hydraulic simulation. The most unstraightforward task in creating the simulation model is to determine the pressure head loss parameters of the pipelines, which requires knowledge of the dimensions of the elbows (i.e. curved pipe segments) [23], and this can only be determined after a 3D pipe routing has been accomplished.

II. LITERATURE REVIEW

Models of engineered systems in which piping is a major component have been developed by process control engineers using conventional approaches, as well as CPS researchers using a multitude of innovative modeling technologies. In applications that claim to be CPSs, pipelines play a major role in building HVAC systems [36], oil pipelines [11], [62], ship chilled water systems [29], industrial tank processes [4] and water supply networks [2], [61], [38]. The used modeling approaches have been selected according to the goals of the CPS research and include complex network theory [61], cross-entropy optimization [11] and linear modeling [2], [4], none of which satisfy our requirement 3 on fidelity in section 1. The term CPS is used in [62], although the physical model of wireless sensor networks for oil pipeline modeling is limited to the sensor networks. CPS testbeds that use a lab setup instead of a model of the physical system [38] are not relevant for the problem formulation in this paper. In summary, the objectives of current research on CPS are different from our work, so conventional modeling approaches in the process industries will be investigated in the remainder of this section.

Since requirement 1 in section 1 states that the desired modeling approach is usable early in the design lifecycle, it is desirable for authors to have positioned their proposed model building activity within some lifecycle model of the development process. Most lifecycle models for industrial plant design are limited to control software development [13], [21], [40]. However, early phase simulation is especially useful if the goal is to optimize performance at the system level rather than the subsystem level [3], in which case a multi-disciplinary lifecycle model should be used such as in [7], [10], [49] supported by multi-disciplinary system specification techniques [51], [52] and architectures [58].

While many articles on new simulation based methodologies are published every month, the great majority of them lacks any mention of an underlying lifecycle model. This is especially true for work in process industries (e.g. [1], [6], [14], [19], [26], [35], [48]). All of these studies involve high fidelity thermo-hydraulic simulations that have been built manually from a P&ID or other source information resembling a P&ID. Although development lifecycle aspects are not elaborated, it is clear that the physical process design will not be modified based on findings from simulations, so simulation results will only be used for control system development. Few works include a partial lifecycle perspective related to the operation time use of the system, but the purpose of simulation is still limited to control system design [30], [34] or fault detection and isolation rather than system design [37], [50]. None of these works adhere to a design philosophy in mechatronic systems that states that control systems and the system to be controlled should be designed concurrently to achieve optimal system level performance [3]. While these works satisfy requirement 3 from section 1, they do not satisfy requirements 1, 2 or 4.

If the control system and the physical process to be controlled were to be designed concurrently, simulations results could result in new requirements for an improved overall system design rather than only an improved control system.

The redesign effort would depend on the stage of the development lifecycle at which these requirements become available. Information on this effort could be used to motivate investment to simulation or rescheduling of activities in the development lifecycle. There is a lack of research about the redesign effort for physical systems and CPS, but the problem is well studied in the software engineering community both in the design lifecycle [12], [20], [60], for the redesign of systems in operation [46] and for re-engineering legacy systems [28]. All of these would be relevant lifecycle cases for industrial plants, but unfortunately there is a lack of scientific literature outside the software engineering field. The software engineering community's experiences and research on redesign eventually led to the agile design lifecycle and superior capabilities for reacting to changing customer needs [56]. A similar agile capability would be very necessary for all design activities along the lifecycle of industrial plants in this era of the "fourth industrial revolution" [59], and the automatic generation of simulation models, which this paper contributes to, is one key capability to this direction [22]. Another such capability would be the concurrent design of the physical process and its control systems, which is discussed in section 6 as further work.

Several works focus on putting industrial plant design information into a machine readable format [9], [15]–[17], [25], [33], [42], paving the way for further research on automatic generation of simulation models, so these are enabling technologies towards satisfying requirements 2 and 4. Research on using such information for actual automatic generation of simulation models is very limited, targeting the generation of low fidelity models from machine readable P&IDs [8] as well as legacy P&IDs using image recognition [5]. An approach using additional machine-readable source information available in a brown-field project is presented in [24]. These works do not attempt to target high fidelity models, so automatic routing of pipelines is not attempted, although it is identified as possible further work in [8]. Thus these works satisfy requirements 1, 2 and 4, but they do not satisfy requirement 3. The positioning of such model generation technology into the design lifecycle has been addressed in [41] and [43]; the latter envisions an integrated use of simulation throughout the plant lifecycle, as per requirements 1 and 2, based on a global survey with industrial and academic participants. A method for higher fidelity model generation from MathML has also been developed [31], but this is a code generator rather than a solution for overcoming the labor-intensive task of simulation model building, so it satisfies requirement 3 without satisfying requirements 1, 2 and 4.

Based on the analysis of the referenced works, it was discovered that none of them targets the full set of requirements 1-4, so the research goal in section 1 targets a gap in the research. In fact, the analysis revealed a broader research gap: no previous work addresses requirements 1 and 3.

Finally, there remains the question of where in the plant lifecycle should the methodology presented in this paper be positioned. As will be discussed in section 6, the further work based on this paper is expected to lead to a novel, concurrent design lifecycle, so it is not possible to reference an existing lifecycle model. The lifecycle presented in [43] is promising, especially since it is motivated by the input of over 200 experts globally, but further work is required to elaborate this lifecycle with more details, before it can be used to position our proposed contribution.

III. PROPOSED ALGORITHM

In order to be able to illustrate the details of the algorithm to the reader, it is assumed that the pipelines are connected to tanks. However, the algorithm is defined in general terms. The pipeline endpoints and the direction of the pipeline at the endpoints are specified in the parameters to the algorithm in Table 1. The naming of these parameters assumes the existence of tanks, in order to improve readability. The algorithm can be generalized simply by renaming these parameters.

TABLE 1. Parameters to the proposed algorithm (vectors in bold).

Type	Name	Description
Vector3f	$\mathbf{conn_A}$	the center of the hole where the pipeline should be connected to tankA
Vector3f	$\mathbf{conn_B}$	the center of the hole where the pipeline should be connected to tankB
boolean	top_A	true if the said hole is at the top of tankA
boolean	top_B	true if the said hole is at the top of tankB

The algorithm will require the use of position, translation and rotation information in three dimensions, so the coordinate systems and directions of rotation defined in Fig. 1 will be used.

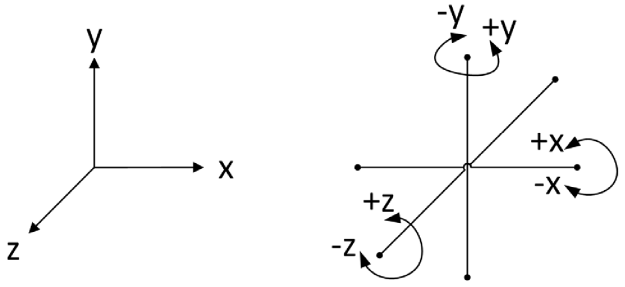

FIGURE 1. Coordinate system (left) and directions of rotations around the three axes (right).

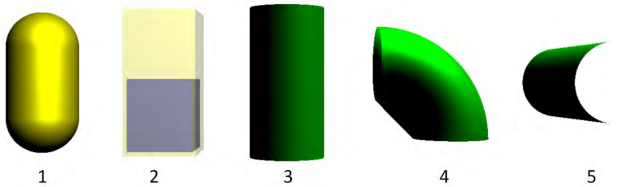

FIGURE 2. Tanks and pipe segments.

Fig. 2 shows the constructs used in this research, which have been generated in the 3D Java environment that will

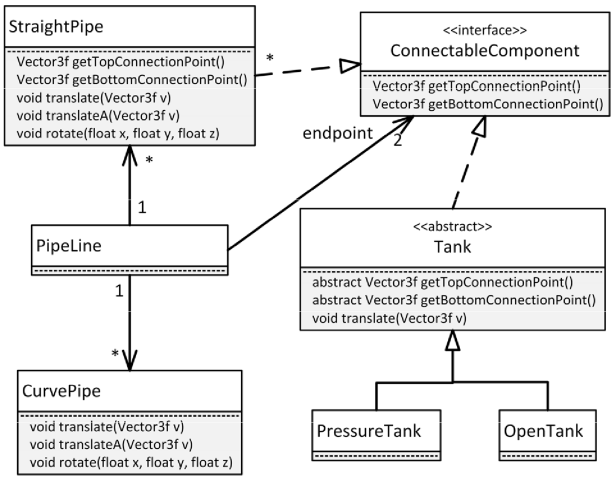

FIGURE 3. UML class diagram that includes the building blocks from Fig. 2.

be used to implement and visualize the proposed algorithm. From left to right they are the following:

1. Pressure tank
2. Open tank (with transparent walls and half full of liquid)
3. Vertical straight pipe segment
4. Curve pipe segment
5. Horizontal straight pipe segment (the mesh used to create the 3D surface is only visible from the outside, creating the impression that half of the pipe is missing – this problem will never be encountered when the pipe segments are connected).

Before presenting an algorithm for creating pipelines between tanks, the constructs in Fig. 2 need to be captured as software entities. This is done in the UML class diagram in Fig. 3, which omits all implementation details that are not essential for presenting the algorithm. Vector3f is a three dimensional vector of float values. The abstract class Tank defines the methods that are used by the algorithm, so new types of tank that implement these methods may be later added without breaking the algorithm. In order to generalize the algorithm beyond pipelines connecting tanks, the interface "ConnectableComponent" is defined to support other kinds of components such as heat exchangers. "StraightPipe" also implements this interface, and in section 5.2 this will be used to let the designer override the default implementation of the algorithm and to specify a location though which the pipeline should be routed. The constructor of StraightPipe has a parameter for choosing between a vertical or horizontal pipe segment illustrated in Fig. 2. StraightPipe and CurvePipe define all the methods to implement the translations and rotations in the pipeline generation algorithm specified in Table 5, Table 6, Table 8 and Table 9. Each segment has two endpoints A and B. The method "translate" translates the center of the pipe segment to the specified 3D position "v", while "translateA" translates endpoint A to "v".

In this paper a 3D pipeline creation algorithm is proposed to create a 3D model of a pipeline connecting two tanks. It is generic for any tanks with top and bottom connection points and it will be demonstrated with the open tank and pressure tank components illustrated in Fig. 2. The pipeline shall be constructed from the components 3-5 in Fig. 2; the straight pipe segments have a length parameter and the curve pipe segment has no parameter. The algorithm is implemented in the PipeLine class in Fig. 3 and it is given the parameters specified in Table 1.

Table 2 defines variables that will be used to specify the proposed algorithm.

TABLE 2. Variables definitions.

Variable	Type	Definition
r	float	radius of the pipe
rise	float	$conn_A.Y - conn_B.Y$
$line_{XZ}$	Vector3f	$\begin{pmatrix} conn_A.X - conn_B.X \\ 0 \\ conn_A.Z - conn_B.Z \end{pmatrix}$
α	float (radians)	arctan($line_{XZ}$.X, $line_{XZ}$.Z)
$pipe_A$	float	length of vertical pipe segment connected to TankA (red segments in Fig. 4)
$pipe_B$	float	length of vertical pipe segment connected to TankB (blue segments in Fig. 4)
$pipe_H$	float	$line_{XZ}/2 - 4r$ (length of horizontal pipe in pipeline with 2 horizontal pipes: Fig. 6 green and yellow segment)

The first task of the algorithm is to determine the length of the straight vertical pipe segments that are connected to the tanks. All the cases that need to be considered in a general solution are shown in Fig. 4, with TankB on the left and TankA on the right.

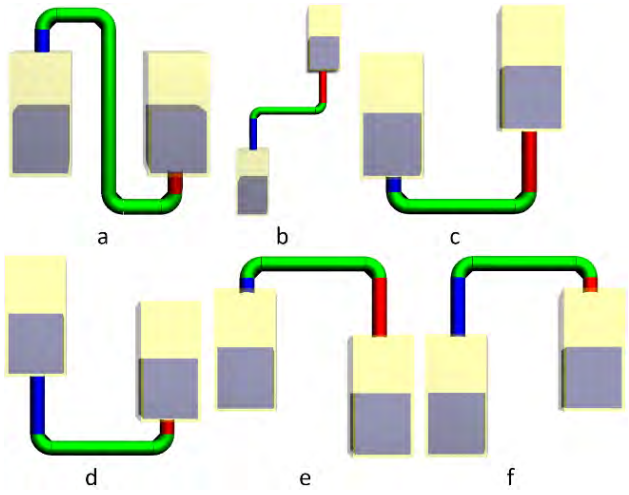

FIGURE 4. Six possible cases for the logic that determines the length of tankAConnectionPipe (red) and tankBConnectionPipe (blue). In each figure, tankA is on the right.

Table 3 shows the descriptions and pipe length parameters for the six cases in Fig. 4.

TABLE 3. Description and pipe length parameters for the cases in Fig. 4.

Case	Description	$pipe_A$	$pipe_B$
a	Connecting top of one tank to the bottom of another. Two horizontal pipes	3r	3r
b	Connecting top of one tank to the bottom of another. One horizontal pipe	(rise+4r)/2	(rise+4r)/2
c	Connecting bottom of tank A to bottom of tank B. Tank B is lower	2r+rise	2r
d	Connecting bottom of tank A to bottom of tank B. Tank A is lower	2r	2r-rise (rise <0)
e	Connecting top of tank A to top of tank B. Tank A is lower	2r-rise (rise<0)	2r
f	Connecting top of tank A to top of tank B. Tank A is higher	2r	2r+rise

Now it is necessary to define the translations and rotations for each pipe segment in order to generate a 3D model of the pipeline. These will be different depending on whether the endpoints of the pipeline are connected to the top or the bottom of tankA and tankB. There are four possible combinations, three of which are depicted in the images a, c, and e of Fig. 4 – the pipeline from the top of TankA to the bottom of TankB is not depicted in Fig. 4. In order to define a single algorithm to handle all of these cases, auxiliary variables are defined in terms of the parameters ''topA'' and ''topB'' from Table 1, since the values of these variables determine which of the four above mentioned cases is in question. Table 4 defines the auxiliary variables ''signA'' and ''zeroA'' derived from ''topA''; ''signB'' and ''zeroB'' are derived analogously from ''topB''. ''signA'' is used to toggle the sign of an expression and ''zeroA'' is used to set an expression to zero.

TABLE 4. Auxiliary variables defined in terms of "top_A" from Table 1.

	top_A = false	top_A = true
$sign_A$	-1	1
$zero_A$	1	0

The pipeline routing has two different cases: a single horizontal pipeline (b-f in Fig. 4) and two horizontal pipelines (a in Fig. 4). According to the values for case a in Table 3, two horizontal pipelines are needed if the vertical distance between pipeline endpoints is less than 6 times the radius of the pipe.

The case for a single horizontal pipeline will be addressed first and the 5 segments of such a pipeline are illustrated in different colors in Fig. 5. In Fig. 5, TankA is above and to the right of Tank B, but the translations and rotations of the segments in Table 5 and Table 6 make no assumptions about the relative positions of the pipeline endpoints in the 3D world. The translations and rotations in Table 5 and Table 6 are defined according to the variables and parameters in Table 1, Table 2, Table 3 and Table 4 according to the coordinate system in Fig. 1. The initial rotations of the pipe segments are depicted in Fig. 2 (to avoid confusion, the camera direction is the same for all the screenshots in this paper).

TABLE 5. Translations of the positions of the pipe segments in Fig. 5 and Fig. 6.

Segment	Translation
red	$\mathbf{conn_A} + \begin{pmatrix} 0 \\ sign_A * pipe_A/2 \\ 0 \end{pmatrix}$
orange	$\mathbf{conn_A} + \begin{pmatrix} 0 \\ sign_A * pipe_A \\ 0 \end{pmatrix}$
green	$\mathbf{conn_A} - \begin{pmatrix} 2r * \sin(\alpha) \\ -sign_A * (pipe_A + 2r) \\ 2r * \cos(\alpha) \end{pmatrix}$
pink	$\mathbf{conn_B} + \begin{pmatrix} 0 \\ sign_B * pipe_B \\ 0 \end{pmatrix}$
blue	$\mathbf{conn_B} + \begin{pmatrix} 0 \\ sign_B * pipe_B/2 \\ 0 \end{pmatrix}$

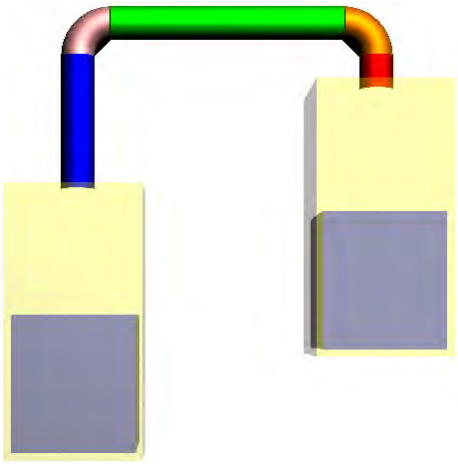

FIGURE 5. Pipe segments in the case of a single horizontal pipe.

The case of a pipeline with 2 horizontal segments is shown in Fig. 6. The translations and rotations of the red, orange, green, blue and pink segments are the same as in Table 5 and Table 6. The translations and rotations of the gray, magenta, cyan, yellow and pink segments are shown in Table 8 and Table 9, using the variable ''pipeV'' defined in Table 7.

IV. IMPLEMENTATION

In order to maximize the application potential of the proposed algorithm, its implementation should support machine-readable source information that is generated in modern plant development projects. There are at least two competing open standards, namely CAEX and extensions of ISO 15926 to the process industries by the DEXPI (Data Exchange in the Process Industry) initiative. There is a growing body of scientific literature on CAEX, and the papers [9], [16], [33], [42] referenced in section 2 are only some examples. There is much less research on ISO 15926 that is relevant to process industries [15], [17], but the standard is supported by major P&ID tool vendors participating in DEXPI (http://www.dexpi.org/). A comparative study of strengths

TABLE 6. Rotations of the pipe segments in Fig.5 and Fig.6 around the axes as defined in Fig. 1.

Segment	Rotation around axes (x,y,z)
red	$\begin{pmatrix}0\\0\\0\end{pmatrix}$
orange	$\begin{pmatrix}\text{zero}_A * \pi\\ -\frac{\pi}{2}+\alpha\\ 0\end{pmatrix}$
green	$\begin{pmatrix}0\\\alpha\\0\end{pmatrix}$
pink	$\begin{pmatrix}\text{zero}_B * \pi\\ \frac{\pi}{2}+\alpha\\ 0\end{pmatrix}$
blue	$\begin{pmatrix}0\\0\\0\end{pmatrix}$

TABLE 7. pipeV: length of magenta pipe in Fig. 6.

	pipe_V
(NOT top_A) AND top_B	2*pipe_A - rise
(NOT top_B) AND top_A	2*pipe_A + rise

TABLE 8. Translation of the positions of pipe segments in Fig.6.

Segment	Translation
gray	conn_A - $\begin{pmatrix}(4r+\text{pipe}_H)*\sin(\alpha)\\ -\text{sign}_A * \text{pipe}_A\\ (4r+\text{pipe}_H)*\cos(\alpha)\end{pmatrix}$
magenta	conn_A - $\begin{pmatrix}(4r+\text{pipe}_H)*\sin(\alpha)\\ \text{sign}_A * (\frac{\text{pipe}_V}{2} - \text{pipe}_A)\\ (4r+\text{pipe}_H)*\cos(\alpha)\end{pmatrix}$
cyan	conn_A - $\begin{pmatrix}(4r+\text{pipe}_H)*\sin(\alpha)\\ \text{sign}_A * (\text{pipe}_V - \text{pipe}_A)\\ (4r+\text{pipe}_H)*\cos(\alpha)\end{pmatrix}$
yellow	conn_A - $\begin{pmatrix}(6r+\text{pipe}_H)*\sin(\alpha)\\ \text{sign}_A * (\text{pipe}_V + 2r - \text{pipe}_A)\\ (6r+\text{pipe}_H)*\cos(\alpha)\end{pmatrix}$

and weaknesses of CAEX and ISO 15926 is provided in [25]. Since a de-facto standard has not yet established itself, this paper defines a lightweight intermediate format for the source information that is required for generating the 3D industrial process model. This format is a text file that contains the very minimum of information that is needed, in order to minimize the cost and effort of exploiting the proposed early phase 3D process model generation technology. The text file may contain two kinds of lines using the following syntax:

```
create <ConnectableComponent> <id> <coordinates>
connect <id1> <interface1> <id2> <interface2>
```

<ConnectableComponent> should be replaced by the name of any class that implements the ''ConnectableComponent'' interface in Fig. 3. <id> is a string identifier that uniquely identifies an instance of this class. <interface> is a string that identifies the mechanical interface of the component to which a pipeline may be connected; in this implementation the possible values are ''top'' and ''bottom''. <coordinates> is a string that specifies the component's location in the 3D space as a list of comma-separated X,Y,Z coordinates. The format is case-insensitive. The snippet below has all the information that the algorithm needs to generate a 3D model of the tanks and the pipelines between them shown in Fig. 11.

TABLE 9. Rotations of the pipe segments in Fig.5 and Fig.6 around the axes as defined in Fig. 1.

Segment	Rotation around axes (x,y,z)
gray	$\begin{pmatrix}\text{zero}_A * \pi\\ \frac{\pi}{2}+\alpha\\ 0\end{pmatrix}$
magenta	$\begin{pmatrix}0\\0\\0\end{pmatrix}$
cyan	$\begin{pmatrix}\text{zero}_B * \pi\\ -\frac{\pi}{2}+\alpha\\ 0\end{pmatrix}$
yellow	$\begin{pmatrix}0\\\alpha\\0\end{pmatrix}$

```
create opentank tank1 6,-6,-20
create opentank tank2 -3,-6,-15
create pressuretank tank3 3,4,-10
connect tank1 bottom tank2 top
connect tank2 bottom tank3 bottom
connect tank3 top tank1 top
```

Competent engineers will be able to generate the required source information into this intermediate format either from standards or from proprietary formats in case the owner of such a format would like to exploit this model generation technology. In this section, an implementation for CAEX is described. The CAEX XML schema is defined in IEC 62424 [27], and it has been adopted also as the top-level format of the AutomationML standard. Thus the free AutomationML editor is used to create the process description of the case study presented in section 5.2. Full details of the CAEX schema are available in the standard [27], but this presentation is aimed at giving the reader sufficient understanding of the schema for the purposes of this discussion. All italic names in this section refer to elements defined in the CAEX schema. CAEX consists of 4 parts: InterfaceClassLib, RoleClassLib, SystemUnitClassLib and InstanceHierarchy. InterfaceClassLib can be used to specify any kind of interfaces, including mechanical interfaces such as holes in tanks into which pipelines may be connected. The following XML snippet has the InterfaceClassLib of the case study, which defines a top and bottom interface type:

```
<InterfaceClassLib Name="InterfaceClassLib1">
  <Version>1.0.0</Version>
  <InterfaceClass
Name="I\_ConnectableComponent\_Bottom"/>
  <InterfaceClass
Name="I\_ConnectableComponent\_Top" />
</InterfaceClassLib>
```

These two interfaces should be included in all component types that are defined in CAEX and from which it is expected to generate classes that realize the ''ConnectableComponent'' interface in Fig. 3. The component types are defined in the CAEX SystemUnitClassLib and the following snippet defines the SystemUnitClass for ''OpenTank''. It instantiates the interfaces defined above and also has an attribute ''position'' which contains the information required for the <coordinates> element in our intermediate format.

```
<SystemUnitClass Name="OpenTank">
 <Attribute Name="position"
AttributeDataType="xs:string" />
  <ExternalInterface Name="bottom"
RefBaseClassPath="InterfaceClassLib1/
I\_Connectable
Component\_Bottom"
ID="8df0d407-58ce-4965-af87-60ddb3cefd83" />
  <ExternalInterface Name="top"
RefBaseClassPath="InterfaceClassLib1/I
\_Connectable
Component\_Top"
ID="8f21fefe-61f4-4ec2-878f-a7004fed35f6" />
</SystemUnitClass>
```

The snippet below is an InternalElement from the InstanceHierarchy and it has all the information from which the ''create opentank tank1 6,-6,-20'' command is created. The ID ''52da53f3-d70f-4f3e-ab8d-181b1b54f9f7'' is a RFC4122 UUID (Universally Unique IDentifier) automatically generated by the AutomationML Editor.

```
<InternalElement Name="tank1"
RefBaseSystemUnitPath="SystemUnitClassLib1/OpenTan
k"
ID="52da53f3-d70f-4f3e-ab8d-181b1b54f9f7">
  <Attribute Name="position"
AttributeDataType="xs:string">
  <Value>6,-6,-20</Value>
 </Attribute>
 <ExternalInterface Name="bottom"
RefBaseClassPath="InterfaceClassLib1/
I\_Connectable
Component\_Bottom"
ID="126484ed-ae14-4e8d-896e-b30a7b447dd6" />
  <ExternalInterface Name="top"\\
RefBaseClassPath="InterfaceClassLib1/
I\_Connectable
Component\_Top"
ID="4f11e6a0-56e5-47be-a6f2-fe9c73109152" />
</InternalElement>
```

Now it is possible to make the linkage in CAEX that corresponds to a pipeline to be generated by the proposed algorithm. The following InternalLink from the InstanceHierarchy has the same information as our intermediate format statement ''connect tank1 bottom tank2 top''. The value of RefPartnerSideA is a string that has been formed by appending the UUID of ''tank1'', a colon and the interface name ''bottom''. Similarly RefPartnerSideB identifies ''tank2'' and its interface ''top''.

```
<InternalLink Name="InternalLink1"
RefPartnerSideA="52da53f3-d70f-4f3e-ab8d-
181b1b54f9f7:bottom"
RefPartnerSideB="64f4cb68-cfcd-4399-a649-
e2570edcbb6e:top"/>
```

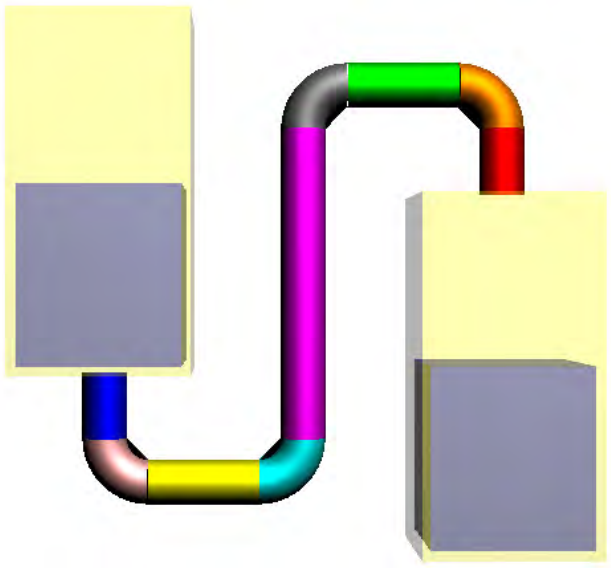

FIGURE 6. Pipe segments in the case of two horizontal pipes.

A XSLT (Extensible Stylesheet Language Transformation) was developed that reads the CAEX file as input and generates the text file in the intermediate format. This text file can be read in the chosen 3D virtualization environment, which in our case is the Java based JMonkeyEngine3. The design in Fig. 3 has been implemented in this environment, and the application will generate the required classes upon parsing this text file. For each line starting with ''create'', the specified class implementing the ''ConnectableComponent'' interface is instantiated; the constructor creates the 3D geometry shown in Fig. 2 and immediately adds it to the 3D model to the scene graph, so these components are visible as soon as the application is executed. Each line starting with ''connect'' has the information required to determine the parameters in Table 1. ''connA'' and ''connB'' are obtained by the ''getTopConnectionPoint()'' or ''getBottomConnectionPoint()'' method, depending on whether the text file specified the ''top'' or ''bottom'' interface. The values for ''topA'' and ''topB'' are true if the text file specifies ''top'' and false otherwise. The constructor for ''PipeLine'' is called with these parameters after which it executes the algorithm in section 3, after which the 3D model is complete.

The classes in Fig. 3 generate the geometries in Fig. 2 from built-in or custom triangular meshes in the 3D Java environment. Another option would have been to import such meshes from CAD models, and such a capability could be worth supporting in future work that targets applying this technology to the detailed design phase when vendor specific models of process components have been selected. However, in the early design phase which is targeted by this research, it is advantageous to be able to flexibly specify the dimensions of the process components and create process models that can be used to assess and develop designs and thus obtain requirements for the selection of vendor specific components. Even after such components have been selected, it is not

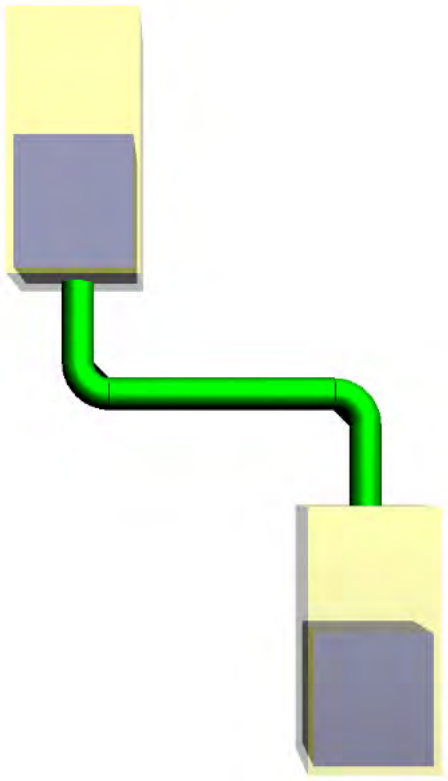

FIGURE 7. Test case for equivalence partition 4 in Table 10.

TABLE 10. Equivalence partitions "P" resulting from considering pipeline endpoint placements at the top of bottom of the tanks and relative heights of the tanks.

P	top_A	top_B	rise	Expected outcome
1	false	true	<6r	2 horizontal pipes (Fig. 4a)
2	false	true	>6r	1 horizontal pipe (Fig. 4b)
3	true	false	>6r	2 horizontal pipes (Fig. 6)
4	true	false	<6r	1 horizontal pipe (Fig. 7)
5	true	true	>0	1 horizontal pipe (Fig. 4f)
6	true	true	<0	1 horizontal pipe (Fig. 4e)
7	false	false	>0	1 horizontal pipe (Fig. 4c)
8	false	false	<0	1 horizontal pipe (Fig. 4d)

TABLE 11. Equivalence partitions "P" for the relative X and Z positions of the tanks.

P	$conn_A.X > conn_B.X$	$conn_A.Z > conn_B.Z$
A	True	True
B	True	False
C	False	True
D	False	False

advantageous to generate the pipe segments from CAD components since attributes such as length and curvature can be flexibly varied directly in the 3D environment. The algorithm in section 3 assumes that the path angle of "CurvePipe" segments is 90 degrees, but some piping specifications also require 45 degree angles, which should be supported in further work.

V. RESULTS

A. VALIDATION

In order to validate the proposed algorithm, a set of test cases with adequate test coverage is required. The number

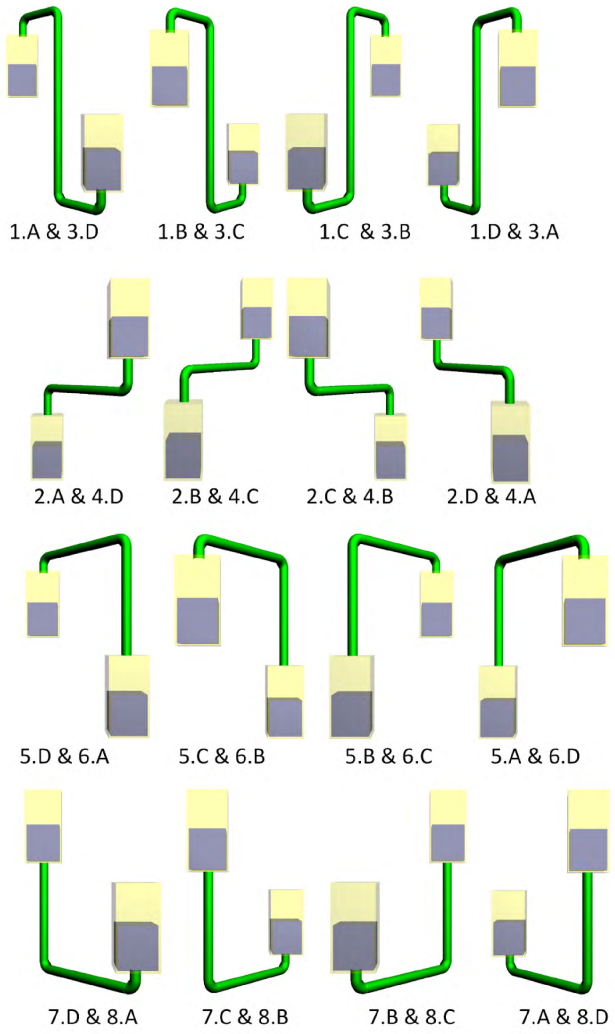

FIGURE 8. Test cases.

of relative positions of two tanks in a 3D world is infinite, so the testing technique of equivalence partitioning [47] is used. An equivalence partition is a range of input values for the system under test, with which the system is specified to behave in a similar way, so it is desirable to have a test case from each partition. The inputs to the algorithm to be tested are the parameters in Table 1, which are used to specify the partitions in Table 10 and Table 11 ("rise" is derived from "connA" and "connB" as specified in Table 2). As an example of equivalence partitioning, consider the case of TankA bottom being connected to TankB top. In section 3 it was stated that if the vertical distance between pipeline endpoints is less than 6 times the radius of the pipe, 2 pipes are needed. Thus one equivalence partition is all tank placements where "rise" is less that "6r" and another is where "rise" is greater than "6r". There should be one test from each partition, and these are illustrated in Fig. 4a and Fig. 4b, respectively, from which the tester can visually confirm that the number of horizontal pipes is indeed according the specifications. Table 9 shows the equivalence partitions resulting from the

FIGURE 9. Case study: laboratory heat production plant.

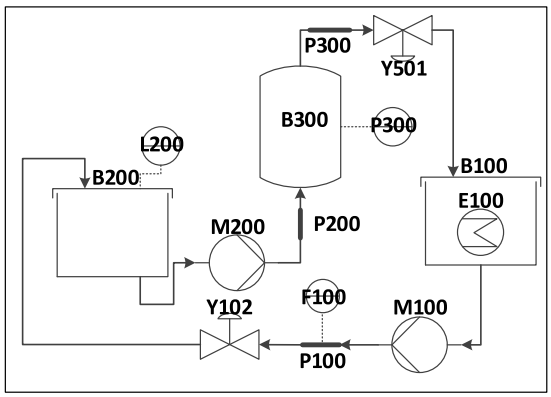

FIGURE 10. P&ID for the primary line (marked with green tape) of the process in Fig.9.

consideration of relative heights of the tanks and placement of the pipeline endpoints at the bottom or top of the tanks.

For each of the cases in Table 10, a figure has been references in the column "expected outcome" to give an example test case belonging to that partition. In each of these figures, both tanks have had the same Z coordinate and the X coordinate of TankA has been greater than that of TankB. (In all screenshots in this paper, the camera is looking into the direction of the negative z axis, so TankA appears to the right of TankB Fig. 4, Fig. 5, Fig. 6, Fig. 7 and Fig. 8.) If the relative X and Z positions of the tanks would be changed, the test coverage would be increased,

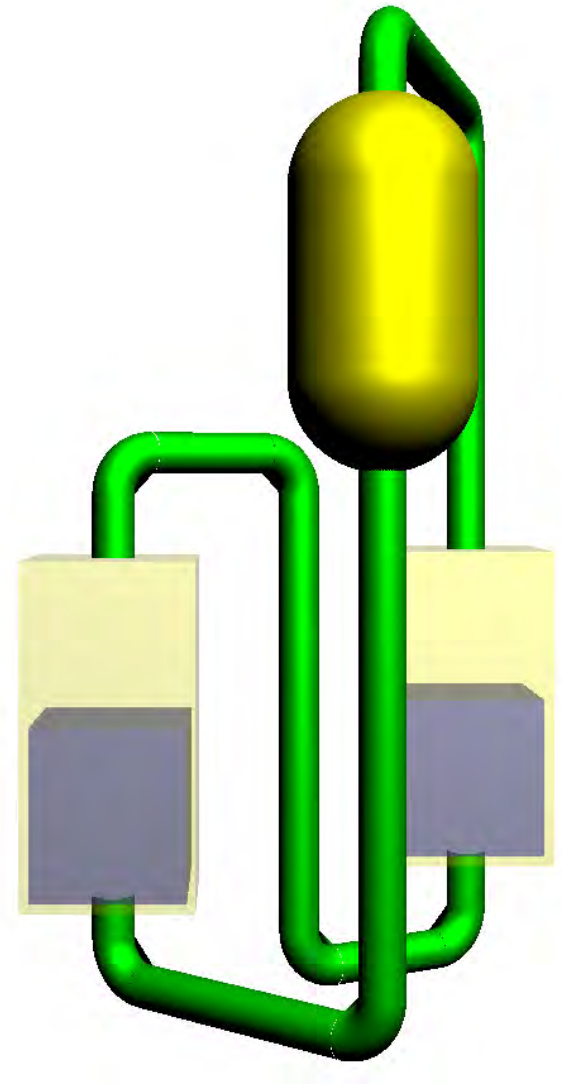

FIGURE 11. Result of applying the proposed algorithm to the process description in Fig.10.

possibly revealing errors in the translation or rotation of pipe segments. Thus, 4 new partitions are defined in Table 11, and since these partitions have been defined independently from the ones in Table 10, all combinations of partitions should be considered, resulting in $8^*4 = 32$ test cases. Test cases identifiers are of the form "X.Y", where "X" is the partition from Table 10 and "Y" is the partition from Table 11. The results from executing these tests are shown in Fig. 8, from which it is possible to visually confirm that the pipelines are according to specifications. Over the course of performing the 32 test cases, it was confirmed that 16 of the test cases are redundant since they result in the same 3D model. Thus, each test case screenshot in Fig. 8 is labelled with the two redundant test cases that both generate the same visual result. For example, 1.A and 3.D give the same result: in 1.A, tankA is the tank in the foreground while in 3.D it is the tank in the background. These cases are equivalent from the perspective of testing the pipeline generation algorithm proposed in this

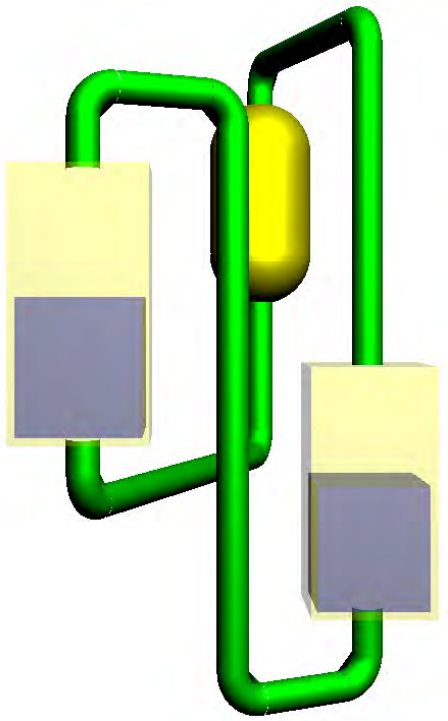

FIGURE 12. Result of applying the algorithm after adjusting tank positions.

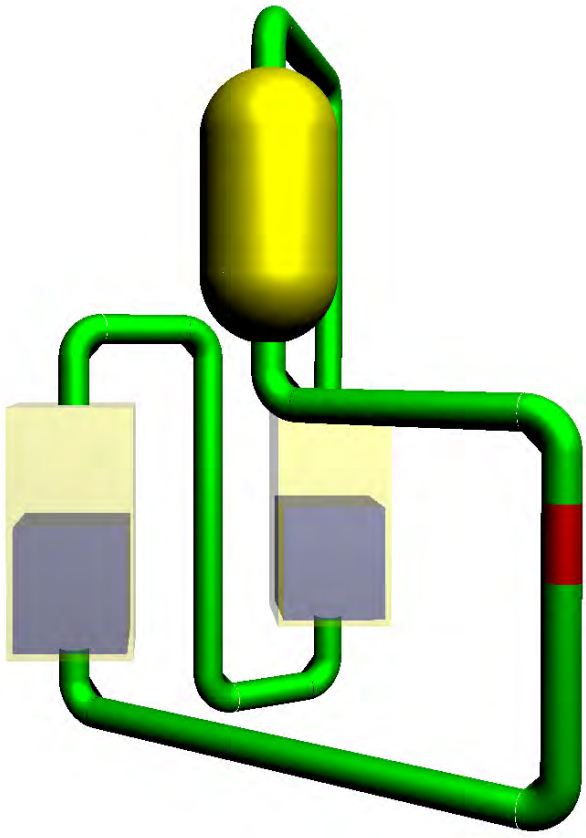

FIGURE 13. Using the StraightPipe (red segment) as a component that implements the ConnectableComponent interface (Fig. 3) to force the pipeline through this segment.

paper, since the algorithm parameters in Table 1 only specify the 3D location and direction of the pipeline at the endpoints without any information about the identity of the component at the endpoint.

B. CASE STUDY

The case study is a laboratory heat production plant at the authors' laboratory (Fig. 9) that was built for investigating novel automation development processes [21]. Specifically, the case study is focused on the primary line of this process which is marked with green tape in Fig. 9. The P&ID of the primary line is in Fig. 10, consisting of three pipelines. The purpose of this case study is to apply the proposed algorithm to automatically generate these pipelines into a 3D model. As discussed in section 4, the tanks and piping related aspects of this P&ID have been expressed in the machine readable CAEX. The 3D position vectors that specify the tank positions cannot be generated from information in the P&ID, so the designer is expected to give initial approximations for the positions into ''position'' parameter described in section 4. The ''getTopConnectionPoint'' and ''getBottomConnectionPoint'' methods of the Tank class in Fig. 3 use the tank dimension parameters to determine the locations of the tank pipeline interfaces based on the tank position, resulting in the values for the ''connA'' and ''connB'' parameters required by the algorithm. The values for the ''topA'' and ''topB'' parameters are set true if the interface type in the CAEX is ''I_ConnectableComponent_Top'' and false otherwise.

As stated in section 1, the primary motivation of the proposed work is not to generate final high fidelity process models. Rather, the purpose is to generate models as early as possible in the development lifecycle, so that the models are accurate enough to inform further system design decisions. The sufficient level of accuracy will be determined by the judgment of the expert process designer. There is no intention to fully automate the process designer but to equip the designer with tools that make it possible to generate the models with minimal additional effort. This should be accomplished so that the organization that employs the designer will not perceive the proposed model generation activity as problematic either from the perspective of additional salary costs or delays in the project schedule.

The following actions are required from the process designer: after creating the initial model in Fig. 11, the designer will visually inspect the model and adjust the tank position coordinates if necessary and then rerun the pipeline generation algorithm. As an example, Fig. 12 shows the result after the leftmost tank has been moved into the direction of the positive Y-axis and the pressure tank has been moved into the direction of the negative Z-axis. This process may be repeated until the process designer declares that the model is suitable for the subsequent design phases in the plant development lifecycle.

Even after adjusting the positions of the tanks, the designer may not be entirely satisfied with the automatically generated

design. For example, in Fig. 11, there may be a need to keep clear the area under the pressure tank and the pipeline should go around it. For this purpose, the ''StraightPipe'' in Fig. 3 also implements the ''ConnectableComponent'' interface, so that the designer may specify such a segment in the CAEX – in Fig. 13 such a segment is marked in red. In the CAEX it is created as an InternalElement just like the tanks with a 3D position attribute, which the designer sets. The designer also uses InternalLinks to connect the top of the segment to the bottom of the PressureTank and the bottom of the segment to the bottom of the leftmost OpenTank. Now the algorithm is forced to route the pipeline though this segment, so that the area under the pressure tank is kept clear of piping.

VI. CONCLUSION

In section 1, the research goal was elaborated with a list of four requirements, which were addressed in the paper as follows:

1. The source information that was required for the algorithm was presented in detail in section 4. This information will be present in a P&ID diagram as soon as the tanks and the pipelines between the tanks have been placed. As discussed in section 5.2 in the context of Fig. 13, the methodology supports other connectable components as well as tanks, so other process components such as heat exchangers could be used instead of tanks. All of this information can be expected to be available early in the design process. Further, a first approximation for the 3D location of all process components should be specified as a triple of X,Y,Z coordinates, and it is reasonable to assume that entering this volume of information will not be a major task for an expert process designer, since they are first approximations that can be easily changed later.
2. The processing of source information as described in section 4 has been automated, so the duration of the model generation activity will depend on how long the process designer will spend on the tasks mentioned in section 5.2. These tasks are the visual inspection of the generated 3D model, updating the process component positions and optionally forcing the routing of pipelines through specific locations as described in Fig. 13. Each of these tasks involves only a few mouse clicks and entering a very limited number of coordinates.
3. As has been elaborated in section 3, the process of generating the pipelines results in detailed information about each pipe segment's elevation, curvature and dimensions, which satisfies the definition of a first-principle model according to [45]. This information would enable building accurate thermo-hydraulic simulations based on correct elevations of nodes and pressure head losses in pipelines between nodes.
4. The overall process designer's workload consists of the tasks that have been explicitly identified under items 1 and 2 above. In current state-of-the-art, first-principle models are created manually [45]; however, that main motivation of this work is not to automate current process design practice. The main advantage of the proposed automatic 3D model generation is to enable new types of rapid prototyping practices, which are currently not feasible due the engineering cost of building process models.

After discussing how the results satisfy the research goals stated in section 1, it remains to discuss the usefulness and application potential of the results and further work. The models generated by the method presented in this paper cannot be simulated, but they contain source information required to build a thermo-hydraulically accurate simulation. Some of this source information is explicit, such as the elevation of the tanks, whereas especially the head loss parameters for entire pipelines need to be computed from several model parameters. Further, the automation related aspects in the P&ID diagram have not been included in the model generation, but the machine readable representations for the source information discussed in section 4 have support also for these aspects. This enables further work on automatic generation of automation functionality into the thermo-hydraulic process simulation, resulting in a CPS simulation as defined in [59]. Thus, the results in this paper open a potential for further work, which will be pursued by the authors, for a highly automatic process of generating high fidelity CPS simulation models early in the lifecycle.

In order to recognize the impact of completing this envisioned further work, it is necessary to appreciate that current design lifecycles consists of separate physical process design and subsequent control system design, as discussed in section 2 with references [1], [6], [14], [19], [26], [35], [48]. Thus, the further work would result in a concurrent design lifecycle aiming at designing an optimal CPS rather than an optimal control system for a given physical process.

REFERENCES

[1] W. A. K. Al-Maliki, F. Alobaid, V. Kez, and B. Epple, ''Modelling and dynamic simulation of a parabolic trough power plant,'' *J. Process Control*, vol. 39, pp. 123–138, Mar. 2016.

[2] S. Adepu, S. Shrivastava, and A. Mathur, ''Argus: An orthogonal defense framework to protect public infrastructure against cyber-physical attacks,'' *IEEE Internet Comput.*, vol. 20, no. 5, pp. 38–45, Sep./Oct. 2016.

[3] J. van Amerongen, ''Mechatronic design,'' *J. Mechatron.*, vol. 13, no. 10, pp. 1045–1066, Dec. 2003.

[4] J. Araujo, M. Mazo, Jr., A. Anta, P. Tabuada, and K. H. Johansson, ''System architectures, protocols and algorithms for aperiodic wireless control systems,'' *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 175–184, Feb. 2014.

[5] E. Arroyo, M. Hoernicke, P. Rodríguez, and A. Fay, ''Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams,'' *Comput. Chem. Eng.*, vol. 92, no. 2, pp. 112–132, Sep. 2016.

[6] A. Balogh, O. M. Aamo, and M. Krstic, ''Optimal mixing enhancement in 3-D pipe flow,'' *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 1, pp. 27–41, Jan. 2005.

[7] G. Barbieri, C. Fantuzzi, and R. Borsari, ''A model-based design methodology for the development of mechatronic systems,'' *Mechatronics*, vol. 24, no. 7, pp. 833–843, Oct. 2014.

[8] M. Barth and A. Fay, ''Automated generation of simulation models for control code tests,'' *Control Eng. Pract.*, vol. 21, no. 2, pp. 218–230, Feb. 2013.

[9] M. Barth, M. Strube, A. Fay, P. Weber, and J. Greifeneder, "Object-oriented engineering data exchange as a base for automatic generation of simulation models," in *Proc. 35th Annu. Conf. IEEE Ind. Electron.*, Porto, Portugal, Nov. 2009, pp. 2465–2470.

[10] L. Bassi, C. Secchi, M. Bonfe, and C. Fantuzzi, "A SysML-based methodology for manufacturing machinery modeling and design," *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 6, pp. 1049–1062, Dec. 2011.

[11] X. Chen *et al.*, "Design automation for interwell connectivity estimation in petroleum cyber-physical systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 2, pp. 255–264, Feb. 2017.

[12] D. Damian and J. Chisan, "An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management," *IEEE Trans. Softw. Eng.*, vol. 32, no. 7, pp. 433–453, Jul. 2006.

[13] G. Doukas and K. Thramboulidis, "A real-time-linux-based framework for model-driven engineering in control and automation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 3, pp. 914–924, Mar. 2011.

[14] D. C. Dumitrache, B. De Schutter, A. Huesman, and E. Dulf, "Modeling, analysis, and simulation of a cryogenic distillation process for ^{13}C isotope separation," *J. Process Control*, vol. 22, no. 4, pp. 798–808, Apr. 2012.

[15] V. Ebrahimipour and S. Yacout, "Ontology-based schema to support maintenance knowledge representation with a case study of a pneumatic valve," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 4, pp. 702–712, Apr. 2015.

[16] S. Faltinski, O. Niggemann, N. Moriz, and A. Mankowski, "AutomationML: From data exchange to system planning and simulation," in *Proc. IEEE Int. Conf. Ind. Technol.*, Athens, Greece, Mar. 2012, pp. 378–383.

[17] X. Fiorentini, T. Paviot, V. Fortineau, J.-L. Goblet, and S. Lamouri, "Modeling nuclear power plants engineering data using ISO 15926," in *Proc. Int. Conf. Ind. Eng. Syst. Manage. (IESM)*, Rabat, Morocco, Oct. 2013, pp. 1–6.

[18] D. E. D. Fuentes *et al.*, "Evaluation and simulation of building automation systems based on their AutomationML description," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Berlin, Germany, Sep. 2016, pp. 1–6.

[19] S. Geist *et al.*, "Dynamic simulation of an integrated thermal separation unit considering practically relevant conditions and limitations," *J. Process Control*, vol. 23, no. 7, pp. 980–989, Aug. 2013.

[20] N. E. Gold, A. M. Mohan, and P. J. Layzell, "Spatial complexity metrics: An investigation of utility," *IEEE Trans. Softw. Eng.*, vol. 31, no. 3, pp. 203–212, Mar. 2005.

[21] D. Hästbacka, T. Vepsäläinen, and S. Kuikka, "Model-driven development of industrial process control applications," *J. Syst. Softw.*, vol. 84, no. 7, pp. 1100–1113, Jul. 2011.

[22] P. Hehenberger, B. Vogel-Heuser, D. Bradley, B. Eynard, T. Tomiyama, and S. Achiche, "Design, modelling, simulation and integration of cyber physical systems: Methods and applications," *Comput. Ind.*, vol. 82, pp. 273–289, Oct. 2016.

[23] R. C. Hibbeler, *Fluid Mechanics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2015, pp. 33–528.

[24] M. Hoernicke, A. Fay, and M. Barth, "Virtual plants for brownfield projects," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Luxembourg, Luxembourg, Sep. 2015, pp. 1–8.

[25] T. Holm, L. Christiansen, M. Göring, T. Jäger, and A. Fay, "ISO 15926 vs. IEC 62424—Comparison of plant structure modeling concepts," in *Proc. IEEE 17th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Kraków, Poland, Sep. 2012, pp. 1–8.

[26] M. Hou, Y. S. Xiong, and R. J. Patton, "Observing a three-tank system," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 3, pp. 478–484, May 2005.

[27] *Representation of Process Control Engineering—Requests in P&I Diagrams and Data Exchange Between P&ID Tools and PCE-CAE Tools*, document IEC 62424, Edition 2.0, IEC International Electrotechnical Commission, 2016, p. 344.

[28] D. Jackson and J. Chapin, "Redesigning air traffic control: An exercise in software design," *IEEE Softw.*, vol. 17, no. 3, pp. 63–70, May 2000.

[29] P. Kadera and P. Novák, "Performance modeling extension of directory facilitator for enhancing communication in FIPA-compliant multiagent systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 688–695, Apr. 2017.

[30] G. D. Kontes, C. Valmaseda, G. I. Giannakis, K. I. Katsigarakis, and D. V. Rovas, "Intelligent BEMS design using detailed thermal simulation models and surrogate-based stochastic optimization," *J. Process Control*, vol. 24, no. 6, pp. 846–855, Jun. 2014.

[31] S. Kuntsche, T. Barz, R. Kraus, H. Arellano-Garcia, and G. Wozny, "MOSAIC a Web-based modeling environment for code generation," *Comput. Chem. Eng.*, vol. 35, no. 11, pp. 2257–2273, Nov. 2011.

[32] T. Kurtoglu and I. Y. Tumer, "A graph-based fault identification and propagation framework for functional design of complex systems," *J. Mech. Des.*, vol. 130, no. 5, pp. 1–8, Mar. 2008.

[33] A. Lüder, N. Schmidt, and R. Rosendahl, "Behavior validation of production systems within different phases of the engineering process," in *Proc. 39th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Vienna, Austria, Nov. 2013, pp. 6906–6911.

[34] Q. Luo, K. B. Ariyur, and A. K. Mathur, "Control-oriented concentrated solar power plant model," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 2, pp. 623–635, Mar. 2016.

[35] W. L. Luyben, "Aspen Dynamics simulation of a middle-vessel batch distillation process," *J. Process Control*, vol. 33, pp. 49–59, Sep. 2015.

[36] M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger, "Intelligent buildings of the future: Cyberaware, deep learning powered, and human interacting," *IEEE Ind. Electron. Mag.*, vol. 10, no. 4, pp. 32–49, Dec. 2016.

[37] R. Mattone and A. D. Luca, "Nonlinear fault detection and isolation in a three-tank heating system," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 6, pp. 1158–1166, Nov. 2006.

[38] E. E. Miciolino, R. Setola, G. Bernieri, S. Panzieri, F. Pascucci, and M. M. Polycarpou, "Fault diagnosis and network anomaly detection in water infrastructures," *IEEE Des. Test*, vol. 34, no. 4, pp. 44–51, Aug. 2017.

[39] H. Nikula, S. Sierla, B. O'Halloran, and T. Karhela, "Capturing deviations from design intent in building simulation models for risk assessment," *J. Comput. Inf. Sci. Eng.*, vol. 15, no. 4, pp. 041011-1–041011-11, Nov. 2015.

[40] M. Obermeier, S. Braun, and B. Vogel-Heuser, "A model-driven approach on object-oriented PLC programming for manufacturing systems with regard to usability," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 790–800, Jun. 2015.

[41] M. Oppelt and L. Urbas, "Integrated virtual commissioning an essential activity in the automation engineering process: From virtual commissioning to simulation supported engineering," in *Proc. 40th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Dallas, TX, USA, Oct./Nov. 2014, pp. 2564–2570.

[42] M. Oppelt, M. Barth, M. Graube, and L. Urbas, "Enabling the integrated use of simulation within the life cycle of a process plant: An initial roadmap: Results of an in-depth online study," in *Proc. IEEE 13th Int. Conf. Ind. Inform. (INDIN)*, Cambridge, U.K., Jul. 2015, pp. 49–55.

[43] M. Oppelt, G. Wolf, and L. Urbas, "Towards an integrated use of simulation within the life-cycle of a process plant," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Luxembourg, Luxembourg, Sep. 2015, pp. 1–8.

[44] N. Papakonstantinou, S. Sierla, D. C. Jensen, and I. Y. Tumer, "Simulation of interactions and emergent failure behavior during complex system design," *J. Comput. Inf. Sci. Eng.*, vol. 12, no. 3, pp. 031007-1–031007-10, Aug. 2012.

[45] C. C. Pantelides and J. G. Renfro, "The online use of first-principles models in process operations: Review, current status and future needs," *Comput. Chem. Eng.*, vol. 51, no. 5, pp. 136–148, Apr. 2013.

[46] M. P. Papazoglou, V. Andrikopoulos, and S. Benbernou, "Managing evolving services," *IEEE Softw.*, vol. 28, no. 3, pp. 49–55, May/Jun. 2011.

[47] R. Patton, *Software Testing*. Indianapolis, IN, USA: Sams, 2005, pp. 67–69.

[48] L. N. Petersen, N. K. Poulsen, H. H. Niemann, C. Utzen, and J. B. Jørgensen, "An experimentally validated simulation model for a four-stage spray dryer," *J. Process Control*, vol. 57, pp. 50–65, Sep. 2017.

[49] D. Regulin, T. Aicher, and B. Vogel-Heuser, "Improving transferability between different engineering stages in the development of automated material flow modules," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1422–1432, Oct. 2016.

[50] V. Reppa, P. Papadopoulos, M. M. Polycarpou, and C. G. Panayiotou, "A distributed architecture for HVAC sensor fault detection and isolation," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 4, pp. 1323–1337, Jul. 2015.

[51] C. Secchi, M. Bonfe, and C. Fantuzzi, "On the use of UML for modeling mechatronic systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 1, pp. 105–113, Jan. 2007.

[52] C. Secchi, M. Bonfé, C. Fantuzzi, R. Borsari, and D. Borghi, "Object-oriented modeling of complex mechatronic components for the manufacturing industry," *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 6, pp. 696–702, Dec. 2007.
[53] S. Sierla, I. Tumer, N. Papakonstantinou, K. Koskinen, and D. Jensen, "Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework," *Mechatronics*, vol. 22, no. 2, pp. 137–151, Mar. 2012.
[54] S. Sierla, B. M. O'Halloran, T. Karhela, N. Papakonstantinou, and I. Y. Tumer, "Common cause failure analysis of cyber–physical systems situated in constructed environments," *Res. Eng. Des.*, vol. 24, no. 4, pp. 375–394, Oct. 2013.
[55] S. Sierla, B. M. O'Halloran, H. Nikula, N. Papakonstantinou, and I. Y. Tumer, "Safety analysis of mechatronic product lines," *Mechatronics*, vol. 24, no. 3, pp. 231–240, Apr. 2014.
[56] D. Spinellis, "Agility drivers," *IEEE Softw.*, vol. 28, no. 4, p. 96, Jul./Aug. 2011.
[57] K. Tahera, C. Earl, and C. Eckert, "A method for improving overlapping of testing and design," *IEEE Trans. Eng. Manag.*, vol. 64, no. 2, pp. 179–192, May 2017.
[58] K. Thramboulidis, "Model-integrated mechatronics—Toward a new paradigm in the development of manufacturing systems," *IEEE Trans. Ind. Informat.*, vol. 1, no. 1, pp. 54–61, Feb. 2005.
[59] A. J. C. Trappey, C. V. Trappey, U. H. Govindarajan, J. J. Sun, and A. C. Chuang, "A review of technology standards and patent portfolios for enabling cyber-physical systems in advanced manufacturing," *IEEE Access*, vol. 4, pp. 7356–7382, 2016.
[60] J. L. de la Vara, M. Borg, K. Wnuk, and L. Moonen, "An industrial survey of safety evidence change impact analysis practice," *IEEE Trans. Softw. Eng.*, vol. 42, no. 12, pp. 1095–1117, Dec. 2016.
[61] Y. Wei and S. Li, "Water supply networks as cyber-physical systems and controllability analysis," *IEEE/CAA J. Autom. Sinica*, vol. 2, no. 3, pp. 313–319, Jul. 2015.
[62] C. Xia, W. Liu, and Q. Deng, "Cost minimization of wireless sensor networks with unlimited-lifetime energy for monitoring oil pipelines," *IEEE/CAA J. Autom. Sinica*, vol. 2, no. 3, pp. 290–295, Jul. 2015.
[63] S. M. Yacoub and H. H. Ammar, "A methodology for architecture-level reliability risk analysis," *IEEE Trans. Softw. Eng.*, vol. 28, no. 6, pp. 529–547, Jun. 2002.

SEPPO ANTERO SIERLA was born in 1977. He received the M.Sc. degree in Technology from the Helsinki University of Technology with the major embedded systems in 2003, The Ph.D degree in Technology in 2007. He received the title of Docent in the field of Software Design for Industrial Automation from the School of Electrical Engineering, Aalto University in 2013. From 2003 to 2011, he was a Research Scientist with Aalto University and the Helsinki University of Technology. He was a Research Assistant and a Software Developer with two companies. Since 2011, he has been the University lecturer with Aalto University, Finland. He has authored or co-authored 35 publications in Web of Science and holds one patent. His research interests include 3-D simulation and virtualization in the manufacturing and process industries. He is a member of the IEEE Industrial Electronics Society (IES) TC Industrial Informatics. He has been a track program committee member in six IES conferences. In 2016, he was a recipient of a Recognition Award as responsible Teacher for the course Automation 1 with the Aalto University School of Electrical Engineering.

TOMMI A. KARHELA was born in 1972. He received the M.Sc. degree in technology from the Tampere University of Technology with the major computational physics in 1996. He received the Ph.D. degree from the Helsinki University of Technology in 2002. From 1996 to 2009, he was a Research Scientist and a Team Leader with the Technical Research Centre of Finland. Since 2009, he was a Research Professor with the Technical Research Centre, Finland. He has also a parallel position as a Professor of Practice with Aalto University. His research interests include computer simulation technology and semantic data modeling in process industry. He is a member of the IEEE Industrial Electronics Society (IES) TC Industrial Informatics. He was a recipient of a VTT award in 2010 on exceptional performance and Finnish Automation Award in 2011 on the with the Simulation Integration Environment–Simantics. He is the Co-Chair of the subcommittee on Simulation.

VALERIY VYATKIN (M'03–SM'04) was born in 1966. He received the Ph.D. degree in applied computer science from the Taganrog State University of Radio Engineering (TSURE), Russia, in 1992, the Dr. Eng. degree in electrical engineering from the Nagoya Institute of Technology, Japan, in 1999, the Dr. Sc. (Eng.) degree in information and control systems from TSURE, in 1999, and the Habilitation degree from the Ministry of Science and Technology of Sachsen-Anhalt, Germany, in 2002. He is on joint appointment as a Chaired Professor (Ämnesföreträdare) of Dependable Computation and Communication Systems, Luleå University of Technology, Luleå, Sweden, and a Professor of Information and Computer Engineering in Automation with Aalto University, Helsinki, Finland. He was a Visiting Scholar with Cambridge University, U.K., and had permanent academic appointments with the University of Auckland, Auckland, New Zealand, Martin Luther University of Halle-Wittenberg, Halle, Germany, and in Japan and Russia. His research interests include dependable distributed automation and industrial informatics; software engineering for industrial automation systems, artificial intelligence, distributed architectures, and multi-agent systems applied in various industry sectors, including smart grid, material handling, building management systems, data centres, and reconfigurable manufacturing. He was a recipient of the Andrew P. Sage award for the best IEEE Transactions paper in 2012. He has been the Chair of the IEEE IES Technical Committee on Industrial Informatics since 2016.