Identifying Bottlenecks in Manufacturing Systems using Stochastic Criticality Analysis

João Bastos*, Bram van der Sanden*, Olaf Donk§, Jeroen Voeten*‡,
Sander Stuijk*, Ramon Schiffelers*†, and Henk Corporaal*
*Eindhoven University of Technology, Eindhoven, The Netherlands
j.p.nogueira.bastos@tue.nl
†ASML, Veldhoven, The Netherlands
‡TNO Embedded Systems Innovation, Eindhoven, The Netherlands
§ICT Group, The Netherlands

***Abstract*—System design is a difficult process with many design-choices for which the impact may be difficult to foresee. Manufacturing system design is no exception to this. Increased use of flexible manufacturing systems which are able to perform different operations/use-cases further raises the design complexity. One important criterion to consider is the overall makespan and associated critical path for the different use-cases of the system. Stochastic critical path analysis plays a fundamental role in providing useful feedback for system designers to evaluate alternative specifications, which traditional fixed-time analysis cannot.**

In this paper, we extend our formal model-based framework, for the specification and design of manufacturing systems, with stochastic analysis abilities by associating a criticality index to each action performed by the system. This index can then be visualized and used within the framework such that a system designer can make better informed decisions. We propose a Monte-Carlo method as an estimation algorithm and we explicitly define and use confidence intervals to achieve an acceptable estimation error. We further demonstrate the use of the extended framework and stochastic analysis with an example manufacturing system.

***Index Terms*—manufacturing systems, criticality analysis, bottleneck identification, formal specification**

I. Introduction

Performance is a key aspect of manufacturing systems, where most designs are driven towards higher productivity (e.g. makespan). In this context, the identification of system bottlenecks is fundamental, since no modification elsewhere on the product flow will lead to an improvement in performance [1]. Therefore, bottleneck analysis provides useful feedback for system engineers to understand how to design or where to improve their systems. In this paper we address the problem of identifying bottlenecks which limit makespan in a manufacturing system, where actions and activities of the system exhibit stochastic execution times.

Comparably, the field of planning, particularly project planning, has addressed similar issues. The problem of analysis of a project plan to understand the expected completion time (makespan) and the criticality of certain tasks (bottleneck analysis) is the same as the one addressed in this paper. Critical path based methods, such as the Critical Path Method (CPM) [2] and the Program Evaluation and Review Technique (PERT) [3], for deterministic and stochastic execution times respectively, have been widely used in this field to address this problem. Moreover, these analysis techniques have found their way into many engineering fields, for example IC design where critical path estimation is used to determine the clock speed, or application scheduling, where critical path analysis is used in scheduling algorithms and heuristics. However, later contributions have shown that better feedback is obtained by the study of the criticality of tasks [4], [5], instead of the criticality of paths (such as the classical critical path methods). These techniques have not yet been applied to manufacturing systems.

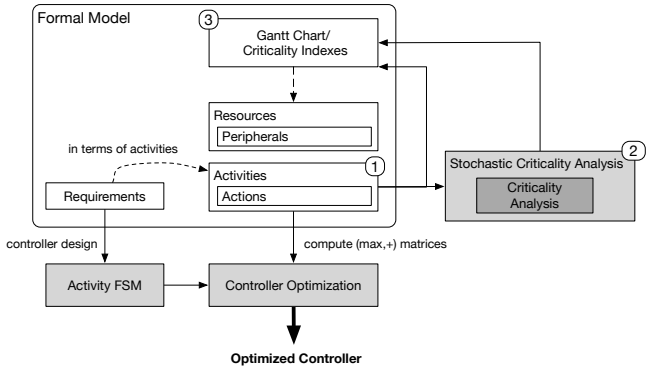

Fig. 1. Overview of the specification framework [6] and extensions.

In this paper we therefore look at the use of criticality of tasks as an analysis technique for bottleneck identification in Manufacturing Systems during design time. For this purpose, we extend our specification framework for Manufacturing Systems [6]. Fig. 1 depicts the overview of the proposed framework [6] and the extensions realized in this paper (denoted by the numbered blocks). Briefly, in this approach the system is specified as a set of resources (e.g. a Drill), containing peripherals (e.g. clamps, motors, heaters) which provide a set of actions. Actions can be used to build activities as Directed Acyclic Graphs (DAGs), where nodes are actions and edges dependencies. We assume no communication delay between tasks. Separately, a set of system requirements can be defined in terms of allowed activity orderings. All these

elements are then used either to synthesize a performance optimal safe controller for the system, or to study a particular desired activity sequence.

To address bottleneck identification for stochastic execution timed activities, we add the following extensions to the framework: 1) we define actions which have a stochastic execution time; 2) we define stochastic notions of path and action criticality, based on the work of [5], and introduce a Monte-Carlo based long-run average [7] approach to determine the bottlenecks in the system; and 3) we extend the Gantt chart visualization of Activity Sequences to express the analysis results in a way which is intuitive and useful for the system designer.

The remainder of this paper is organized as follows. We start by discussing the related work in Section II. The formal framework is briefly introduced in Section III. In Section IV, we introduce stochastic timed actions as an extension to the framework, and in Section V we define criticality indices as our metric. Section VI introduces a long-run method for stochastic criticality analysis. Sections VII and VIII discuss the results and their interpretation, and compare to classical critical path analyses. Finally, Section IX discusses the industrial application and Section X concludes the work.

II. Related Work

The use of critical path based methods has made its way into different fields such as embedded systems and IC design, for deterministic execution times [8]–[10] as well as for stochastic execution times [11], [12]. Here the focus is mostly on task scheduling and not on the prediction of performance bottlenecks as in our work.

The study of criticality of a path, or a task in a path, has it origins in the Critical Path Method (CPM) [2] and in the Project Review and Evaluation Technique (PERT) [3], [13]. Projects are modeled as activity networks, where nodes represent tasks, annotated with (deterministic (CPM) or stochastic (PERT)) durations. Edges model dependencies between tasks and a critical path determines the total completion time of the project. In these approaches two measures were defined: critical path and critical task.

Although both CPM and PERT focus on the criticality of paths, is has been shown that is it more meaningful to rely on the Criticality Index (CI) metric [4], [5], [14], [15]. Intuitively, the goal of CPM or PERT is to give feedback on which tasks to focus on minimize the project lead time. The CI metric allows the ranking of the tasks within a project given the likelihood that they are on a critical path. We follow the Criticality Index approach in this paper.

Determination of the CI can be done either: 1) analytically [14], [16], [17] or 2) by Monte-Carlo estimation approaches [5], [15], [18]. For the general case, analytical solutions are too computationally demanding and current solutions cannot handle even medium-sized stochastic activity networks. A review of these approaches is made in [19]. To obtain scalable analysis, our approach will also rely on simulation.

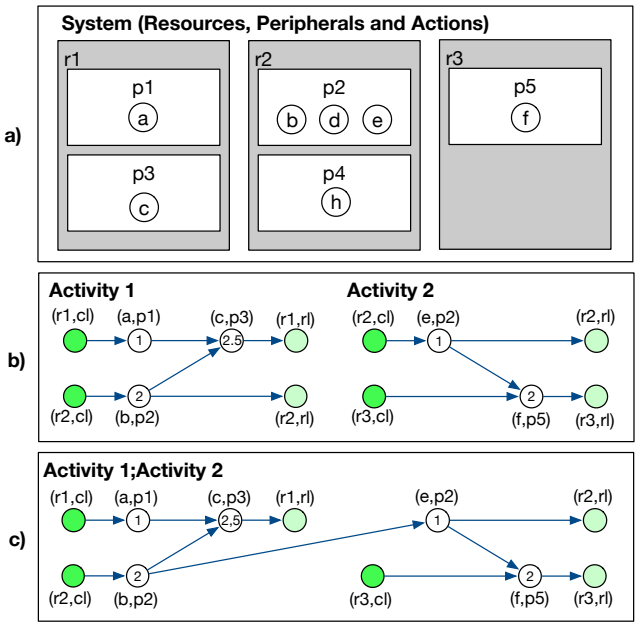

Fig. 2. Symbolic example of a system specification using our framework [6]: a) depicts the resources, peripherals and actions provided, b) two examples of activities constructed with the provided peripheral actions and c) an example of an activity sequence (Activity 1; Activity 2).

Our contribution is the formalization of the stochastic notions of path and action criticality as an extension to our framework [6] together with an effective stochastic critically analysis technique supporting design-time bottleneck identification in Manufacturing Systems.

III. Previous Work

In [6], we define our formal modeling framework for compositional specification of both the functional and timing aspects of manufacturing systems. In this paper we extend this framework with the notion of *stochastic timed actions*. To make the paper self-contained we briefly introduce the basics of the framework.

A. System and Activities

The following sets define the basic elements in the framework:

- set $\mathcal{A}$ of actions, with typical elements $a \in \mathcal{A}$;
- set $\mathcal{P}$ of peripherals, with typical elements $p \in \mathcal{P}$;
- set $\mathcal{R}$ of resources, with typical elements $r \in \mathcal{R}$.

We assume a function $R : \mathcal{P} \to \mathcal{R}$, such that $R(p)$ is the resource that contains p.

Using these elements we can construct DAGs of actions executed by peripherals (or release and claims of resources). Such a DAG represents a particular functional aspect of the system, e.g. movement of a robot arm, handover of products between resources, etc. We will call such a DAG an *activity*.

Definition 1. *An activity is a DAG $(N, \to)$, consisting of a set N of nodes and a set $\to \subseteq N \times N$ of dependencies. We write a dependency $a \to b$, to denote that $(a, b) \in \to$. We assume*

a mapping function $M : N \to \mathcal{A} \times \mathcal{P} \cup \mathcal{R} \times \{rl, cl\}$ *which associates a node to either a pair* (a, p) *(referring to an action executed on a peripheral) or to a pair* (r, v) *with* $v \in \{rl, cl\}$ *(referring to a* claim *(cl) or* release *(rl) of resource* r*). Nodes mapped to a pair* (a, p) *are called* action nodes*, and nodes mapped to a claim or release of a resource are called* claim *and* release *nodes respectively.*

Fig. 2 depicts the model of a manufacturing system. In Fig. 2 a) three resources r_1, r_2 and r_3 are depicted. The individual peripherals of each resource are denoted in the figure by $p_1, .., p_5$. Each peripheral provides a set of actions that can be executed by that peripheral. These actions are denoted by a, b, c, d, e, f and h.

Fig. 2 b) shows two possible activities constructed from the example system of Fig. 2 a). The shaded circles denote claim and release nodes. The open circles depict the actions to be executed by their respective peripherals. Dependencies are denoted by arrows.

Activities have to satisfy a number of static consistency constraints. They are as follows:

1) All nodes mapped to the same peripheral are sequentially ordered to avoid self-concurrency.
2) Each resource is claimed no more than once.
3) Each resource is released no more than once.
4) Every action node is preceded by a claim node on the corresponding resource.
5) Every action node is succeeded by a release node on the corresponding resource.
6) Every release node is preceded by a claim node on the corresponding resource.
7) Every claim node is succeeded by a release node on the corresponding resource.

B. Activity Sequences

Activities can be combined using the activity sequencing operator.

Definition 2 (Sequencing Operator)**.** *Given two activities* $Act_1 = (N_1, \to_1)$ *and* $Act_2 = (N_2, \to_2)$ *with* $N_1 \cap N_2 = \emptyset$*, we define* $Act_1 \cdot Act_2$ *as activity* $Act_{1\cdot 2} = (N_{1\cdot 2}, \to_{1\cdot 2})$*.*

Let $R_{1\cap 2} = R(Act_1) \cap R(Act_2)$ *denote the set of resources used in both activities. Define the set of corresponding release nodes in* N_1*, and claim nodes in* N_2 *as* $rl_{1\cap 2} = \{n_1 \mid n_1 \in N_1 \wedge (\exists r \in R_{1\cap 2} \mid M(n_1) = (r, rl))\}$*, and* $cl_{1\cap 2} = \{n_2 \mid n_2 \in N_2 \wedge (\exists r \in R_{1\cap 2} \mid M(n_2) = (r, cl))\}$ *respectively.*

Activity $Act_{1\cdot 2} = (N_{1\cdot 2}, \to_{1\cdot 2})$ *is now defined as follows:*

$$
\begin{aligned}
N_{1\cdot 2} &= (N_1 \cup N_2) \backslash (cl_{1\cap 2} \cup rl_{1\cap 2}) \\
\to_{1\cdot 2} &= \{(n_i, n_j) \mid\ n_i \to_1 n_j \wedge\ n_j \notin rl_{1\cap 2}\} \cup \\
&\quad \{(n_i, n_j) \mid\ n_i \to_2 n_j \wedge\ n_i \notin cl_{1\cap 2}\} \cup \\
&\quad \{(n_1, n_2) \mid\ (\exists n_{rl} \in rl_{1\cap 2} \mid n_1 \to_1 n_{rl}) \wedge \\
&\qquad\qquad (\exists n_{cl} \in cl_{1\cap 2} \mid n_{cl} \to_2 n_2)\}.
\end{aligned}
$$

An example of the application of the sequence operator in practice is depicted in Fig. 2 c). It depicts the activity resulting from the sequencing of activities A_1 and A_2 in Fig. 2 b). Notice that the resulting activity is formed by coalescing and subsequently removing corresponding release and claim nodes.

IV. Stochastic timed activities

To study the dynamic behavior we need to describe the timing behavior of all actions and their influence on the execution of other actions (Extension 1 in Fig. 1).

In order to model the timing variations that a system exhibits, we attribute to each action a random variable representing the random execution time of that action. To this end we assume $\mathcal{T}$ to denote the collection of all such random variables. We assume that this set contains random variable $\underline{0}$ which takes value 0 with probability 1.

Definition 3 (Stochastic execution time of an action)**.** *We assume a function* $E : \mathcal{A} \to \mathcal{T}$ *that maps each action to its corresponding random execution time variable.*

Definition 4 (Stochastic execution time of a node)**.** *Given activity* $(N, \to)$ *and node* $n \in N$*, we define the execution time of node n as:*

$$
E(n) = \begin{cases} E(a) & \text{if } M(n) = (a, p) \\ & \text{for some } a \in \mathcal{A}, p \in \mathcal{P} \\ \underline{0} & \text{otherwise.} \end{cases}
$$

In order to define the start and end time of each node in an activity, we need to consider the dependencies of all nodes first. To this end, we formally define the set of predecessors of each node.

Definition 5 (Predecessor nodes)**.** *Given activity* $(N, \to)$ *and node* $n \in N$*, we define the set of predecessor nodes:*

$$Pred(n) = \{n_{in} \in N \mid n_{in} \to n\}.$$

An activity also requires information considering the state of the system at the start of its execution. We introduce this as a resource time-stamp function $\gamma_R : \mathcal{R} \to \mathcal{T}$. For each $r \in \mathcal{R}$, $\gamma_R(r)$ represents the current stochastic availability time of resource r at the starting time of the activity. We now define the start and end time of the execution of each node in an activity.

Definition 6 (Start and completion time of a node)**.** *Given activity* $Act = (N, \to)$ *and resource time stamp function* γ_R*, we define the random variables start time* $S(n)$ *and finishing time* $F(n)$ *for each node* $n \in N$*:*

$$
S(n) = \begin{cases} \gamma_R(r) & \text{if } M(n) = (r, cl) \\ & \text{for some } r \in \mathcal{R}, cl \in \{rl, cl\} \\ \max\limits_{n' \in Pred(n)} F(n') & \text{otherwise} \end{cases}
$$

$$F(n) = S(n) + E(n)$$

TABLE I
ACTION EXECUTION TIMES PER CASE

Action	Execution Time		
	Best-Case	Worst-Case	Most Likely-Case
a	1	3	2
b	1.8	2.2	2
c	2.5	4	3.5
e	0.8	1.2	1
f	1.8	2.2	2

TABLE II
NODE CRITICALITY PER CASE

Action	Criticality of Nodes		
	Best-Case	Worst-Case	Most Likely-Case
a	0	1	1
b	1	0	0
c	0	1	1
e	1	0	0
f	1	0	0

V. CRITICAL PATH AND CRITICAL NODE

Now that we have defined stochastic timing behavior of an activity we can introduce the concepts of critical path and critical node.

Definition 7 (Path and Makespan of a Path)**.** *Given an activity* $Act = (N, \rightarrow)$ *we let* Δ *denote the set of all paths of nodes* $\underline{n} = n_1 \cdots n_k$ *such that* $n_i \rightarrow n_{i+1}$ *for each* $i : (1 \leq i < k)$ *and* $n_k \not\rightarrow$*. For each path* $n_1 \cdots n_k \in \Delta$*, we define makespan as:*

$$mks(\underline{n}) = F(n_k) \tag{1}$$

The makespan of a path represents the total time necessary to complete the execution of all peripheral actions in that path. A path for which the makespan is larger or equal to any other path in Δ, and for which the start time of every node is the same as the finishing time of its predecessor is called a *critical path*.

Definition 8 (Critical Path)**.** *For each* $\underline{n} = n_1 \cdots n_k \in \Delta$*, we define a random variable:*

$$C(\underline{n}) = \begin{cases} 1 & \text{if } mks(\underline{n}) = max_{\underline{n'} \in \Delta} mks(\underline{n'}) \\ & \text{and } S(n_{i+1}) = F(n_i) \text{ for } 1 \leq i < k-1 \\ 0 & \text{otherwise} \end{cases}$$

Hence for each path $\underline{n} \in \Delta$, $C(\underline{n})$ indicates whether $\underline{n}$ is a critical path or not. Notice that $C(\underline{n})$ is a Bernoulli-distributed random variable.

A critical path imposes to an activity the total time necessary (makespan) to complete all peripheral actions in that activity. Note that there can be multiple paths which are critical. This would however imply that all those critical paths have the same makespan.

Definition 9 (Critical Node)**.** *For each* $n \in N$*, we define a random variable:*

$$C(n) = \begin{cases} 1 & \text{if } n = n_i \text{ for some path } \underline{n} = n_1 ... n_k \in \Delta, \\ & \text{with } 1 \leq i < k \text{ and } C(\underline{n}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Therefore, for each node $n \in N$, $C(n)$ indicates whether n is in a critical path or not. Note that n can be in multiple paths.

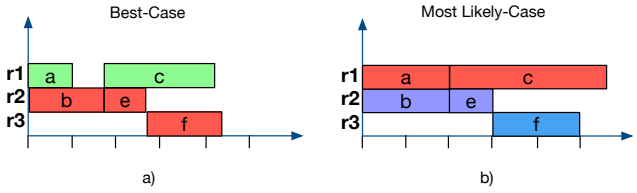

Fig. 3. Different execution scenarios of activity sequence of Fig. 2 c) and in red the critical path: a) for best-case execution times, and b) for most likely-case execution times.

Fig. 3 a) depicts the Gantt chart of the best-case execution of the activity sequence in Fig. 2 c). The actions in red depict the critical path of this execution scenario and the execution times are shown in Table I. In this case the critical path is b e f and nodes b, e and f are critical nodes.

Depending on varying execution times, an activity exhibits different possible critical paths. For example, consider the alternative most likely-case execution of activity sequence of Fig. 2 c), where peripheral actions a and c have, respectively, an execution time of 2 and 3.5, as shown in Table I c). The new resulting Gantt chart is depicted in Fig. 3 b). Now, the critical path is a c and a and c are critical nodes. Table II shows the criticality of a node, according to Definition 9, for each node in the different execution cases.

VI. STOCHASTIC CRITICALITY ANALYSIS

To determine the bottlenecks of a system we propose a stochastic analysis that estimates the criticality of the nodes of an activity (Extension 2 in the Fig. 1). The metric used is the *Criticality Index* (CI) [4], [5] of a node. The Criticality Index is defined as the probability with which a node occurs in the critical path.

Definition 10 (Criticality Index)**.** *Given an* $Act = (N, \rightarrow)$*, the Criticality Index* $c(n)$ *of a node* $n \in N$ *is the probability that the node will be in a critical path. This index is defined as* $c(n) = E(C(n))$*.*

In this section we discuss the approach to estimate the criticality index of a node based on the calculation of confidence intervals. For simplicity of notation, we fix n and will write C to denote $C(n)$ and c to denote $c(n)$. Thus $c = \mathbb{E}(C)$. Furthermore the variance of C is given by $c(1-c)$.

TABLE III
COMPARISON OF PERT APPROACH AND STOCHASTIC CRITICALITY ANALYSIS.

	PERT		SCA
Action	Expected Execution Time	PERT Node Criticality	Criticality Indices (CI)
a	2	1	0.52
b	2	0	0.48
c	3.4	1	0.72
e	1	0	0.28
f	2	0	0.28

A. Estimation of c

To estimate c we define the point-estimator $\widehat{C} = \frac{1}{k}\sum_{i=1}^{k} C_i$, where each C_i represents an independent copy of C. Then by the strong law of large numbers $\widehat{C}$ converges strongly to c. Hence for sufficiently large k, $\frac{1}{k}\sum_{i=1}^{k} c_i \approx c$, where each c_i represents a sample from C_i. We will write $\widehat{c}$ to denote point-estimation $\frac{1}{k}\sum_{i=1}^{k} c_i$.

B. Confidence Intervals and determining k

We would like to determine k such that the absolute error $|c-\widehat{c}|$ is sufficiently small. For this we use the central limit theorem stating that $\widehat{C} = \frac{1}{k}\sum_{i=1}^{k} C_i$ is approximately Normally distributed with expected value kc and variance $kc(1-c)$. Normalization yields $\frac{k\widehat{C}-kc}{\sqrt{kc(1-c)}}$ to be $N(0,1)$ distributed. This also holds when c in the denominator is replaced by point-estimator $\widehat{C}$. Rewriting yields that $\sqrt{k}\frac{\widehat{C}-c}{\sqrt{\widehat{C}(1-\widehat{C})}} \sim N(0,1)$ for sufficiently large k.

Therefore $\mathbb{P}(-Z_{\frac{\gamma+1}{2}} \leq \sqrt{k}\frac{k\widehat{C}-kc}{\sqrt{\widehat{C}(1-\widehat{C})}} \leq Z_{\frac{\gamma+1}{2}}) \approx \gamma$ for confidence level γ and quantile $Z_{\frac{\gamma+1}{2}}$. (Here quantile $Z_p = \Phi^{-1}(p)$, where Φ is the cumulative distribution function of N(0,1)). From this it follows that $\mathbb{P}(c \in [\widehat{C} - Z_{\frac{\gamma+1}{2}}\frac{\sqrt{\widehat{C}(1-\widehat{C})}}{\sqrt{k}}, \widehat{C} + Z_{\frac{\gamma+1}{2}}\frac{\sqrt{\widehat{C}(1-\widehat{C})}}{\sqrt{k}}]) \approx \gamma$.

Replacing $\widehat{C}$ with point-estimation $\widehat{c}$ delivers confidence interval $[\widehat{c} - Z_{\frac{\gamma+1}{2}}\frac{\sqrt{\widehat{c}(1-\widehat{c})}}{\sqrt{k}}, \widehat{c} + Z_{\frac{\gamma+1}{2}}\frac{\sqrt{\widehat{c}(1-\widehat{c})}}{\sqrt{k}}]$ which contains c with confidence γ.

In case c is contained in this interval $|c-\widehat{c}| \leq Z_{\frac{\gamma+1}{2}}\frac{\sqrt{\widehat{c}(1-\widehat{c})}}{\sqrt{k}}$. We use this inequation to stop the simulation once the absolute error is smaller than a predetermined bound.

In practice, for a number k of runs, samples $e_1(n), ..., e_k(n)$ from copies $E_1(n), E_2(n), ..., E_k(n)$ from $E(n)$ are taken, for every node $n \in N$. For each run m, such that $1 \leq m \leq k$, the critical path is computed, following the methodology in [5], [20], using samples $e_m(n)$ of each node. Then values $c_1(n), ..., c_k(n)$ are computed according to Definition 9, together with the error bound just described. The simulation terminates once the absolute error of the estimation of $c(n)$ for every node is smaller than or equal to a specified bound.

VII. INTERPRETATION OF THE CRITICALITY INDEX

In this section we provide an interpretation for the Criticality Index and discuss how it provides more information than classical critical path analyses. For this purpose we use as an example activity sequence of Fig. 2 c).

Let us assume that the action execution times of the activity sequence are modeled by stochastic random variables. Classical critical path analysis, such as the PERT approach [3], consider distinct cases for action execution times, which correspond to the maximal, minimal and mode values of the distribution associated with the random variable. These are depicted as worst (w), best (b) and likely (l) cases in Table I. The PERT approach first computes the expected time for each action, using equation $(b + 4 \times l + w)/6$, and then computes expected critical path with those values. The expected action execution times and the resulting values of the criticality of nodes, according to Definition 9, are depicted in Table III, under column PERT. From these, we conclude that the critical nodes are a and c, and therefore reducing the execution of these actions would result in a lower makespan.

Consider the same example and assumptions used for the PERT approach, but now applying the proposed stochastic criticality analysis. As a result we obtain the Criticality Index (CI) of every node, as in Definition 10. The resulting values are depicted in Table III, under column SCA. The stochastic analysis results indicate that, even though action c has the highest criticality, actions a and b have an almost identical probability of occurring in the critical path. Which therefore implies that the initial conclusion from the PERT analysis is not precise enough and improving action b should also be considered. The advantage of the stochastic criticality analysis is that it considers the shape, and therefore the variance of the distribution associated with the random variables, and the influence of the execution of actions on the execution of dependent actions. This allows for a more complete overview of the criticality of the activity sequence which cannot be obtained by classical critical path approaches.

This is further illustrated in the next section with a manufacturing system case study.

VIII. CASE STUDY: TWILIGHT SYSTEM

The Twilight System [6] is an example created for the study of controller synthesis and performance analysis of Manufacturing Systems. This manufacturing system is a simplification of the product handling model that has been created at ASML[1] the world-leading manufacturer of lithography systems, using similar kinds of peripherals and resources.

In this section we elaborate on the functionality of the Twilight System and on the identification of performance bottlenecks to reduce makespan by using stochastic criticality analysis. Since bottleneck improvement is always possible (there will always be a bottleneck), we insist the (expensive) Drill resource to be the performance limiting resource.

A. Example Manufacturing System

Our example system contains four resources. First, there are two robots to transport balls; the load robot (LR) and the

[1] www.asml.com

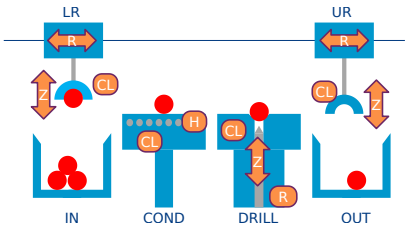

Fig. 4. Manufacturing system example (Twilight system) with two robots and two production stages.

TABLE IV
SET OF ACTIVITIES FOR OUR CASE STUDY.

LR_PickFromInput	LR_PutOnDrill	UR_PutOnCond
LR_PutOnCond	UR_PickFromDrill	UR_PutOnOutput
LR_PickFromCond	UR_PickFromCond	Condition
LR_PickFromDrill	UR_PutOnDrill	Drill

unload robot (UR). Each robot has a homing position; LR on the left corner, and UR on the right corner. The other two resources are processing stations, the conditioner (COND) to ensure a right ball temperature, and the drill (DRILL) to drill a hole in a ball. Both robots have three peripherals; a clamp (CL) to pick up and hold a ball, an R-motor (R) to move along a rail, and a Z-motor (Z) to move the clamp up and down. Both processing stations have a clamp peripheral. The conditioner has a heater (H), to heat a ball. The drill has an R-motor (R) to rotate the drill bit, and a Z-motor (Z) to move the drill bit up and down.

Each ball processed by the system follows the same life cycle. First, a ball is picked up at the input buffer by the load robot. Then it is brought to the conditioner and processed. Next, the item is transported by either one of the robots to the drill, where it is drilled. Finally, the drilled ball is transported to the output buffer by the unload robot.

B. Activities

In our system, there are two activities that process balls: Condition and Drill. For transportation of the balls, there are two types of activities: picking up a ball by a robot, and releasing a ball by a robot on a product location. The complete set of activities is shown in Table IV.

Each activity is modeled formally by specifying the actions involved and the dependencies between these actions. As an example, consider activity LR_PickFromCond shown in Fig. 5, in which the load robot picks a ball from the conditioner.

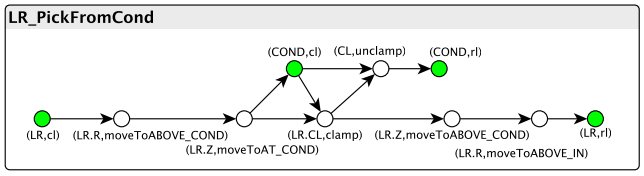

Fig. 5. Activity LR_PickFromCond.

TABLE V
SET OF DISTRIBUTIONS FOR OUR CASE STUDY.

Action	min	max	mode	λ
Clamp/Unclamp	0.2	0.4	0.25	8
LR Moves	5.0	6.2	6	8
UR Moves	4	6	5	8
Condition	4	16	11	5
Drill	4	26	10	5

TABLE VI
CRITICALITY OF NODES AS RESULT OF THE PERT ANALYSIS.

Actions	B	W	L	PERT
Conditioner.CD	0	0	0	0
Drill.ZR (1,2)	0	1	0	0
LR.XY (1,2)	1	0	1	1
LR.XY (3,4)	1	0	1	1
UR.XY (1,2)	0	0	0	0

C. Stochastic Time Modeling

To model the stochastic variability of the nodes in our example system, we model their timing behavior using PERT Distributions [21]. The PERT distribution (also called beta-PERT) is a smooth version of the uniform distribution or triangular distribution. It is characterized by: the minimal value, the maximal value, the mode and a shape parameter λ.

The distributions for the nodes are defined in Table V. Regarding the variability in the system we assume the following: actions performed by clamp peripherals and by the Load and Unload robot exhibit low variance, and the Condition and Drill actions exhibit high variance. Variation in the execution time of the Conditioning action is due to the difference in the initial temperature of the input balls. Finally, the variation exhibited by the Drill action is due to the assumption that depending on the desired depth and type of drilling profile each ball may have different drilling execution times. We further define the activity sequence to be studied: LR_PickFromInput; LR_PutOnCond; Condition; UR_PickFromCond; UR_PutOnDrill; Drill; UR_PickFromDrill; UR_PutOnOutput.

D. Analysis and Results

We implemented the proposed stochastic critical path analysis (Section V) and classical critical path analysis within the framework [6]. We specified the Twilight Systems with the framework (Section III). Visualization of the results is done by coupling with the TRACE viewer [10]. For this case study we assume a confidence interval of 99% and an absolute error $|c - \hat{c}| < 0.001$ for the estimation of the Criticality Indices.

In order to provide the feedback in a more intuitive way, we use colored Gantt charts (Extension 3 in Fig. 1). Fig. 6 depicts the Gantt chart of the results of the PERT analysis, assuming worst case action execution times (left Gantt chart) and the Gantt chart of the stochastic criticality analysis (right Gantt

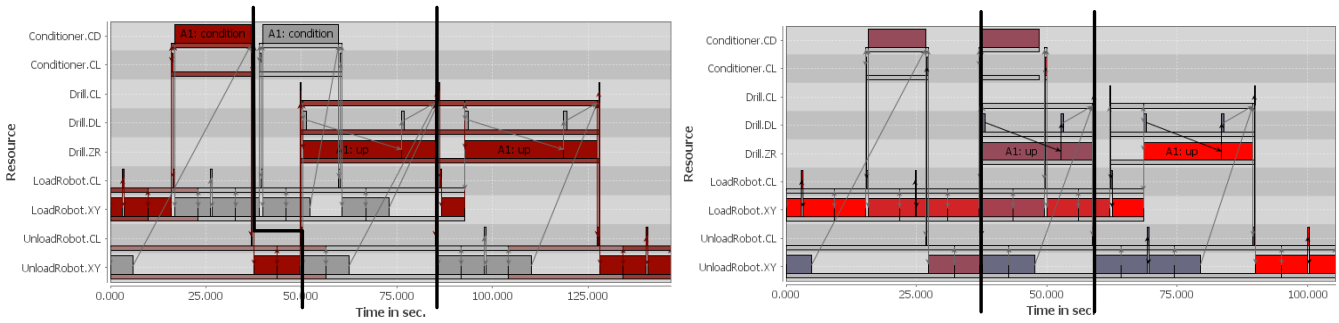

Fig. 6. (Left) Result of classical criticality analysis, assuming worst-case action execution times; (Right) Result of stochastic criticality analysis.

TABLE VII
CRITICALITY INDICES AS RESULT OF THE CRITICALITY ANALYSIS.

Actions	CI	Abs. Error
Conditioner.CD	0.3221	0.0009
Drill.ZR (1,2)	0.3312	0.0009
LR.XY (1,2)	0.3467	0.0008
LR.XY (3,4)	0.6821	0.0008
UR.XY (1,2)	0.0000	0.0008

chart). In both Gantt charts, the shades of red of each node (represented by a block in the Gantt chart) imply a different criticality index for the action. The lighter the redness the lower the criticality index, and vice-versa. For simplicity, we discuss in detail the results within the bounded area (denoted by the black lines). The values of the results of both analysis for the actions within the bounded are provided in Tables VI and VII.

Table VI shows the resulting node criticality values of the classical PERT approach, for best (B), worst (W), most likely (L) and expected (PERT) cases for the action execution times. (Expected execution times are computed using equation $(B+4\times L+W)/6$). Table VII depicts the Criticality Indices of peripheral actions of the case study, resulting from the stochastic criticality analysis, and the associated absolute error of the estimation.

Observing the two tables, it can be concluded that the computed expected case of the PERT approach does not yield a correct overview of the criticality of the actions, when compared to the stochastic criticality analysis results. Observe that according to the expected case results, the Conditioner has a criticality of 0, and the only actions in the critical path are the LR actions. Consequently, to reduce the overall makespan one would improve the execution times of the LR actions. However, the stochastic criticality analysis results contradict this, since the Conditioner action has a CI of 0.3221, i.e. the probability of the Conditioner action being on the critical path is almost one third. Therefore, further improvements to the makespan of the system can be achieved by improving the execution of the Conditioner actions.

Furthermore, even if one would consider all the three cases, best, worst, and most-likely, still the information is not complete. Observe that in none of the three cases the actions of Conditioner occur in the critical path according to the PERT approach. Whilst the stochastic critical analysis indicates a CI of 0.3221. Given this insight, a choice can be made to improve the performance of the Conditioner resource and the Load Robot such that the only remaining bottleneck is the expected and desired, the Drill resource.

IX. INDUSTRIAL APPLICATION

At ASML, the extended framework and stochastic analysis is used to study the performance of different designs of industrial-size models of lithography systems. The framework allows domain engineers to describe the system in terms of resources, peripherals, actions, and activities and easily identify bottlenecks as feedback for design decisions. Furthermore, for a given design, it is possible to study the impact of varying execution times of individual actions, or system use-cases, on the overall system makespan.

X. CONCLUSION

This paper extends our formal modeling approach and framework such that given a compositional specification of both functionality and timing of a manufacturing system, a reliable and realistic stochastic criticality analysis can be performed. In this approach, a Monte-Carlo simulation technique (including error estimation based on confidence intervals) is used to estimated the criticality index of each peripheral action of the system. This analysis provides better and more accurate results than the typical fixed-timed critical path analysis used for bottleneck design. Furthermore, the extended framework is a good fit for the field of manufacturing system and therefore the possibility of using stochastic criticality analysis within the framework further empowers the formal-model based specification and analysis of manufacturing systems.

There are a number of steps to extend the current analysis and framework. For example the addition of sensitivity analysis, where we analyze how each peripheral action affects the overall throughput of the system in terms of its variance and mean deviations, as in [4]. Furthermore, a feedback loop can be designed within the framework to allow for the implementation of design corrections based on the results of the analysis, in a co-aided design approach for the specification of manufacturing systems.

REFERENCES

[1] L. Li, Q. Chang, J. Ni, G. Xiao, and S. Biller, "Bottleneck detection of manufacturing systems using data driven method," in *2007 IEEE International Symposium on Assembly and Manufacturing*, July 2007, pp. 76–81.

[2] J. E. Kelley, Jr and M. R. Walker, "Critical-path planning and scheduling," in *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, ser. IRE-AIEE-ACM '59 (Eastern). New York, NY, USA: ACM, 1959, pp. 160–173.

[3] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar, "Application of a technique for research and development program evaluation," *Operations Research*, vol. 7, no. 5, pp. 646–669, 1959.

[4] S. E. Elmaghraby, "On criticality and sensitivity in activity networks," *European Journal of Operational Research*, vol. 127, pp. 220–238, 2000.

[5] R. M. Van Slyke, "Letter to the editor—monte carlo methods and the pert problem," *Oper. Res.*, vol. 11, no. 5, pp. 839–860, Oct. 1963.

[6] B. van der Sanden, J. a. Bastos, J. Voeten, M. Geilen, M. Reniers, T. Basten, J. Jacobs, and R. Schiffelers, "Compositional specification of functionality and timing of manufacturing systems," in *FDL 16*, 2016.

[7] B. D. Theelen, J. P. M. Voeten, and Y. Pribadi, "Accuracy analysis of long-run average performance metrics," in *STW Technology Foundation, Utrecht (The Netherlands*, 2001, pp. 261–269.

[8] P. Barford and M. Crovella, "Critical path analysis of tcp transactions," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 238–248, Jun. 2001.

[9] P. Bjorn-Jorgensen and J. Madsen, "Critical path driven cosynthesis for heterogeneous target architectures," in *Hardware/Software Codesign, 1997. (CODES/CASHE '97), Proceedings of the Fifth International Workshop on*, Mar 1997, pp. 15–19.

[10] M. Hendriks, J. Verriet, T. Basten, B. Theelen, M. Brassé, and L. Somers, "Analyzing execution traces: critical-path analysis and distance analysis," *International Journal on Software Tools for Technology Transfer*, pp. 1–24, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10009-016-0436-z

[11] Y. Zhan, A. J. Strojwas, M. Sharma, and D. Newmark, "Statistical critical path analysis considering correlations," in *Proceedings of the 2005 IEEE/ACM International Conference on Computer-aided Design*, ser. ICCAD '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 699–704. [Online]. Available: http://dl.acm.org/citation.cfm?id=1129601.1129701

[12] M. Teixeira, R. Lima, C. Oliveira, and P. Maciel, "A stochastic model for performance evaluation and bottleneck discovering on soa-based systems," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, Oct 2010, pp. 358–365.

[13] K. R. MacCrimmon and C. A. Ryavec, "An analytical study of the pert assumptions," *Operations Research*, vol. 12, no. 1, pp. 16–37, 1964.

[14] B. M. Dodin and S. E. Elmaghraby, "Approximating the criticality indices of the activities in pert networks," *Management Science*, vol. 31, no. 2, pp. 207–223, 1985.

[15] R. A. Bowman, "Efficient estimation of arc criticalities in stochastic activity networks," *Management Science*, vol. 41, no. 1, pp. 58–67, 1995.

[16] V. G. Kulkarni and V. G. Adlakha, "Markov and markov-regenerative pert networks," *Operations Research*, vol. 34, no. 5, pp. 769–781, 1986.

[17] J. J. Martin, "Distribution of the time through a directed, acyclic network," *Operations Research*, vol. 13, no. 1, pp. 46–66, 1965. [Online]. Available: http://dx.doi.org/10.1287/opre.13.1.46

[18] C. Sigal, A. Pritsker, and J. Solberg, "The use of cutsets in monte carlo analysis of stochastic networks," *Mathematics and Computers in Simulation (MATCOM)*, vol. 21, no. 4, pp. 376–384, 1979.

[19] M.-J. Yao and W.-M. Chu, "A new approximation algorithm for obtaining the probability distribution function for project completion time," *Computers and Mathematics with Applications*, vol. 54, no. 2, pp. 282 – 295, 2007.

[20] J. E. Kelley, Jr and M. R. Walker, "Critical-path planning and scheduling," in *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, ser. IRE-AIEE-ACM '59 (Eastern). New York, NY, USA: ACM, 1959, pp. 160–173. [Online]. Available: http://doi.acm.org/10.1145/1460299.1460318

[21] C. E. Clark, "Letter to the editorthe pert model for the distribution of an activity time," *Operations Research*, vol. 10, no. 3, pp. 405–406, 1962. [Online]. Available: http://dx.doi.org/10.1287/opre.10.3.405