ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MASTER UNIVERSITARIO EN INGENIERIA INDUSTRIAL

"INVESTIGATION OF VALIDATION METHODS FOR THE EVENT-DRIVEN SIMULATION OF MRO PROCESSES"

Autor:Ignacio Marchena
Tutor: Alexander Georgiadis
Director: Prof.Dr.-Ing.Berend Denkena

Madrid

Septiembre 2016

UNIVERSIDAD PONTIFICIA ICAI ICADE COMILLAS MADRID

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN ACCESO *ABIERTO* (*RESTRINGIDO*) DE DOCUMENTACIÓN

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Ignacio Marchena González , como estudiante de la UNIVERSIDAD PONTIFICIA COMILLAS (COMILLAS), **DECLARA**

que es el titular de los derechos de propiedad intelectual, objeto de la presente cesión, en relación con la obra *„Investigation of validation methods for the event-driven simulation of MRO processes"* que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual como titular único o cotitular de la obra.

En caso de ser cotitular, el autor (firmante) declara asimismo que cuenta con el consentimiento de los restantes titulares para hacer la presente cesión. En caso de previa cesión a terceros de derechos de explotación de la obra, el autor declara que tiene la oportuna autorización de dichos titulares de derechos a los fines de esta cesión o bien que retiene la facultad de ceder estos derechos en la forma prevista en la presente cesión y así lo acredita.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad y hacer posible su utilización de *forma libre y gratuita* (*con las limitaciones que más adelante se detallan*) por todos los usuarios del repositorio y del portal e-ciencia, el autor **CEDE** a la Universidad Pontificia Comillas de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra (a) del apartado siguiente.

3º. Condiciones de la cesión.

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia, el repositorio institucional podrá:

(a) Transformarla para adaptarla a cualquier tecnología susceptible de incorporarla a internet; realizar adaptaciones para hacer posible la utilización de la obra en formatos electrónicos, así como incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato. .

(c) Comunicarla y ponerla a disposición del público a través de un archivo abierto institucional, accesible de modo libre y gratuito a través de internet.[1]

(d) Distribuir copias electrónicas de la obra a los usuarios en un soporte digital. [2]

4º. Derechos del autor.

El autor, en tanto que titular de una obra que cede con carácter no exclusivo a la Universidad por medio de su registro en el Repositorio Institucional tiene derecho a:

a) A que la Universidad identifique claramente su nombre como el autor o propietario de los derechos del documento.

b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

c) Solicitar la retirada de la obra del repositorio por causa justificada. A tal fin deberá ponerse en contacto con el vicerrector/a de investigación (curiarte@rec.upcomillas.es).

d) Autorizar expresamente a COMILLAS para, en su caso, realizar los trámites necesarios para la obtención del ISBN.

d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

---

[1] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría redactado en los siguientes términos:

(c) Comunicarla y ponerla a disposición del público a través de un archivo institucional, accesible de modo restringido, en los términos previstos en el Reglamento del Repositorio Institucional

[2] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría eliminado.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

a) Deberes del repositorio Institucional:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.

- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusiva del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.

- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.

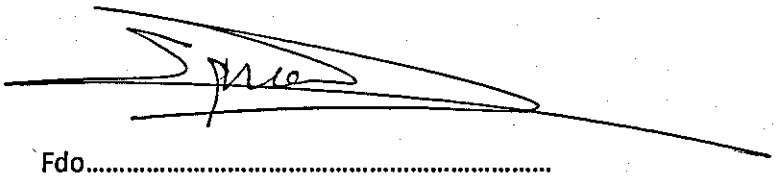
b) Derechos que se reserva el Repositorio institucional respecto de las obras en él registradas:

- retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

UNIVERSIDAD PONTIFICIA ICAI ICADE COMILLAS MADRID

Madrid, a 08 de Septiembre de 2016

ACEPTA



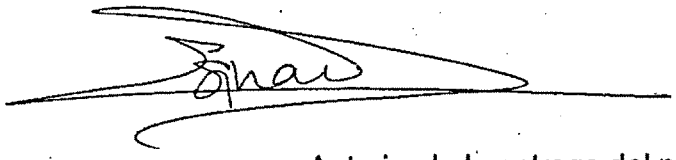
Fdo.....

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

"Investigation of validation methods for the event-driven simulation of MRO processes"

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 15/16. es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

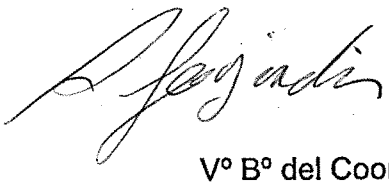
Fdo.: Ignacio Marchena Fecha: 08/ 09/ 2016



Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Alexander Georgiadis Fecha: 14 / 09 / 2016



V° B° del Coordinador de Proyectos

Fdo.: Jaime de Rábago Fecha: ...../ ...../ .....

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MASTER UNIVERSITARIO EN INGENIERIA INDUSTRIAL

"INVESTIGATION OF VALIDATION METHODS FOR THE EVENT-DRIVEN SIMULATION OF MRO PROCESSES"

Autor:Ignacio Marchena
Tutor: Alexander Georgiadis
Director: Prof.Dr.-Ing.Berend Denkena

Madrid

Septiembre 2016

INVESTIGATION OF VALIDATION METHODS FOR THE EVENT-DRIVEN SIMULATION OF MRO PROCESSES

Autor: Ignacio Marchena

Director: Alexander Georgiadis

RESUMEN DEL PROYECTO

1. Introducción

La compañía „“DB Fahrzeuginstandhaltung GmbH” o DB Vehicle Maintenance, tiene desarrollados en el software Tecnomatix: Plant Simulation, diferentes procesos de mantenimiento con la intención de mejorar dichos procesos a través de experimentos o solventar problemas que pudieran presentarse. En este contexto se sitúa la necesidad de validar los resultados de dichas simulaciones, puesto que actualmente la compañía elige un número alto aleatorio y posteriormente analiza los resultados con Excel resultando por tanto en un proceso no automatizado y muchas veces tedioso. El presente trabajo analiza diferentes métodos de cálculo del número de repeticiones necesarias para alcanzar un cierto grado de confianza estadística, más tarde se presenta una metodología para aplicar dos de los métodos analizados así como una implementación en el software de los mismos. Finalmente, los métodos son validados en Excel y se realizan experimentos para valorar el impacto de los mismos y extraer conclusiones.

1.1 Estado del arte

En el estado del arte del proyecto se estudian no solo los métodos relativos a la simulación sino todo el proceso para aportar una visión del conjunto al trabajo. El apartado comienza con descripciones de los tipos de capital que se emplean en industrias dedicadas a MRO que, debido a su complejidad, coste, número de interdependencias y número de partes involucradas justifican la necesidad de realizar un “Planning and control” de las reparaciones de las mismas. En la figura 1, se puede apreciar la diferencia entre procesos de ensamblajes simples y complejos [Bake09] [Kis06] [Marc10] [Den16].

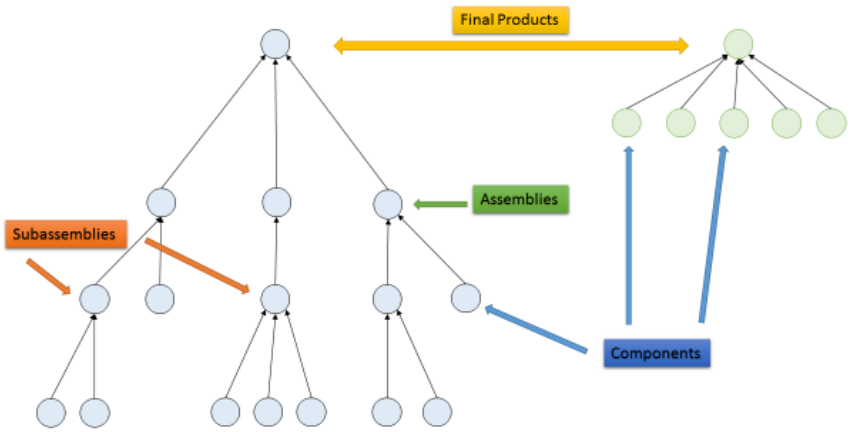


Figura 1

A raíz de las citadas características, se nace la necesidad de las simulaciones. Estas permiten realizar experimentos sin coste añadido o sin parar la los procesos de mantenimiento [Car14] [VRS12]. [Ross16] [Bang10].

1.2 Objeto del Proyecto

El presente proyecto tiene como objetivo la investigación de métodos para validar resultados dados en simulaciones de tipo “Discrete even-driven”. Este tipo de simulaciones se caracterizan porque cada acción sucede en un momento concreto del tiempo y cambia el estado del sistema. Aunque el proceso real podría ser tomando como infinito, las simulaciones realizadas son finitas y es por tanto ese campo de las simulaciones sobre el que se investiga. El siguiente objetivo del proyecto es la implementación de dos métodos en el software Plant Simulation sobre el proceso de reparación de sets de ruedas ya desarrollado por DB.

2. Metodología

Para abordar el problema se dividió en 4 etapas, la primera parte, donde se ha realizado una investigación y resumen de las tecnologías presentes relacionadas con el campo en el que se desarrollaron las simulaciones y otras que podrían ser aplicadas en un futuro con algunos cambios y consideraciones. La segunda parte consta de la metodología para la implementación de los métodos elegidos. En ella se analizan las variables de entrada “inputs” y de salida “outputs ” necesarios para cada uno de los métodos que serán implementados posteriormente asi como los flujos de información en dichos métodos. La implementación es llevada a cabo en una tercera fase y por ultimo se lleva a cabo una validación en Excel asi como un experimento en el que se comparan distintos tipos de simulación para distintos valores de niveles de confianza y error en alguna varaible.

3. Resultados

Los métodos implementados en el software fueron el t-iterative, el half-width method y el ratio method. Los 3 están basados en intervalos de confianza sobre la media, y la diferencia entre ambos radica en la serie de suposiciones que hacen. Tras la implementación de los mismos, se observa que el t-iterative da unos resultados exactos en cuanto al número de simulaciones necesarias. Itera hasta que el grado de confianza deseado es alcanzado. Los otros dos métodos, no dan un valor 100% exacto, sin embargo son útiles ya que permiten establecer una aproximación del número de iteraciones necesarias a partir de una muestra inicial. Al realizar la validación de los métodos se comprobó que solo el t-iterative y el half-width dan resultados válidos. Se considera que un resultado del half-width son válidos ya que en general dan valores mayores que los necesarios sin embargo cabe mencionar que este método tiene una gran dependencia de los valores iniciales que aparezcan en la simulación, y el rango donde estén los mismos.

Por último, se realizó un experimento con los dos métodos considerados validos en los que se prueban para distintas configuraciones de niveles de confianza. En la Figura 2, aparecen el número de simulaciones necesarias para cada variable medida y para cada grado de confianza dados unos valores de error asumidos por la compañía.

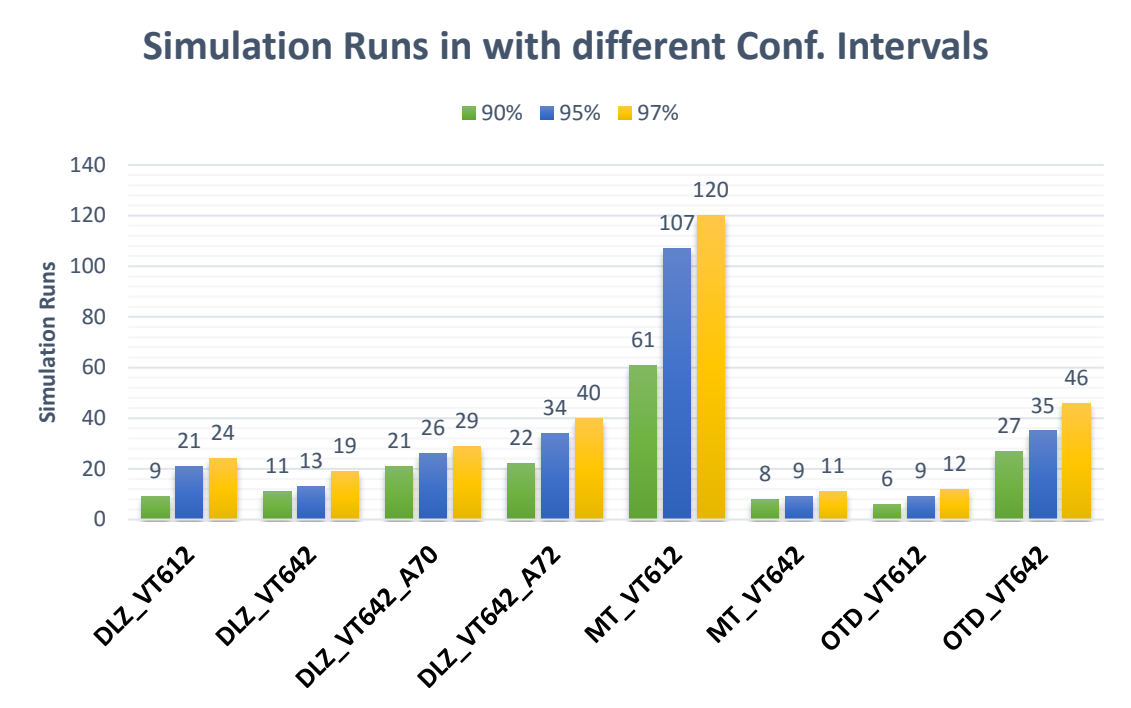


Figura 2

4. Conclusiones

Dada la clara diferencia existente entra la variable MT_VT612, que corresponde al tiempo medio de tardanza del modelo VT612, y el resto de variables analizadas se propone que se analice el modelo para el límite de la segunda variable con mayor número de simulaciones (OTD_VT642). Con este valor, 46 simulaciones, conseguimos una confianza del 97% en todas las variables salvo en la anteriormente mencionada. Otra posible opción sería aumentar el error que se asume en la variable permitiendo así que el deseado grado de confianza se alcance antes.

En relación al otro método válido implementado, half-width method, la conclusión extraída es que es de utilidad si lo que se busca no es tanto una precisión exacta en la validación del resultado como una aproximación del número de simulaciones necesarias. En la última versión de la implementación se acoplan ambos métodos de tal manera que a partir del número inicial de simulaciones fijadas, tenemos una estimación del número necesario gracias al half-width method; este número nos permite decidir en primera instancia cuando puede ser recomendable detener la simulación si la reducción del error en relación a hacer una simulación más es pequeña.

5. Bibliografia

[Bake09] Baker K.R. und Trietsch D. (2009): *Principles of Sequencing and Scheduling*.

[Bang10] Bangsow S. (2010): *Manufacturing Simulation with Plant Simulation and SimTalk*. Springer.

[Car14] Carolin Kellenbrink, Felix Herde, Steffen C. Eickemeyer, Thorben Kuprat, Peter Nyhuis (2014): Planning the regeneration processes of complex capital goods.

[Den16] Denkena B.; Dittrich M.A. und Georgiadis A. (2016): Combining in-house pooling and sequencing for product regeneration by means of event-driven simulation. In: *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering*..

[Kis06] Kister T. und Hawkins B. (2006): Historical View of Maintenance. In *Maintenance Planning and Scheduling: Streamline Your Organization for a Lean Environment*. S. 1-18.

[Marc10] Resico M.F. (2010): *Introduccion a la Economía Social de Mercado*.

[Ross16] Rossetti M.D. (2016): Analyzing Simulation Output. In Wiley (Hg.), *Simulation Modeling and Arena*. 2nd. Aufl. John Wiley & Sons, Inc. S. 300-384.

[VRS12] V.R.S. Raju, O.P. Gandhi, and S.G. Deshmukh (2012): Maintenance, Repair and Overhaul Performance Indicators for Military Aircrafts. *Defence Science Journal*.

INVESTIGATION OF VALIDATION METHODS FOR THE EVENT-DRIVEN SIMULATION OF MRO PROCESSES

Author: Ignacio Marchena

Tutor: Alexander Georgiadis

SUMMARY OF THE THESIS

1. Introduction

The firma „“DB Fahrzeuginstandhaltung GmbH” o DB Vehicle Maintenance, have developed in the software Tecnomatix: Plant Simulation different processes aimed to improve the performance of them through experiments or just to solve problems that could appear eventually. Is in this context where the project find the objective of validate the results of those simulations. In the current state, the researchers of the company choose a high random number of simulation runs and later on analyse the results on excel files. Thus is a process far from being automatic. The present dissertation analyses different methods to compute the necessary number of simulation runs to achieve a certain degree of statistical confidence, later on, a methodology to implement two of those methods as well as the implementation itself in the software is deeply explained. Finally the methods are validated on excel and some experiments are performed in order to value the impact of the changes and bring some conclusions to light.

1.1 State of the art

In the current state of the technology, the methods related to simulation plus the whole process of MRO, are investigated in order to provide a ground for the work. The sections starts with description of the kinds of capital goods used in MRO industries that due to its complexity, investment cost, high number of interdependences and high number of parts involved on them justify the necessity of carry out planning and scheduling stage of the capital good that will be repaired. In the figure 1, we can see the different between simple and complex assembly processes [Bake09] [Kis06] [Marc10] [Den16].

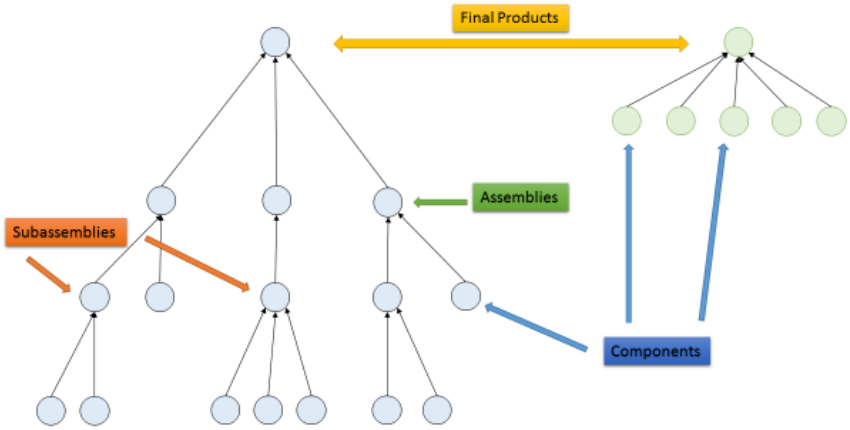


Figure 1

The necessity to carry out simulations arise from the above mentioned characteristics of the goods. The simulations allow the companies to perform experiments without added cost and without stopping the maintenance processes [Car14] [VRS12]. [Ross16] [Bang10].

1.2 Objective of the Project

The work is aimed towards investigation of validation methods for discrete event-driven simulations. In this kind of simulations, each action happen at a specific point in time and change the state of the system. The process faced in the thesis is somehow infinite since new and new parts to repair can come in the future. However, the simulation model considers only one year of repairs, and therefore the methods investigated are focused on finite simulations. The next target is the implementation of two methods in the software Plant Simulation in the smart wheel set maintenance process developed by DB.

2. Methodology

To face the problem, it has been divided into four stages. In the first one, we carry out an investigation and summary of the technologies related with the field on which the simulations take part and others that could be applied in the future under some considerations. The second part contains the methodology for the implementation of the chosen methods. On it, the inputs, outputs and information flows are analysed. The implementation of the methods are developed in a third phase of the project and finally the validation is made in excel files.

3. Results

The methods chosen to be implemented in the software were “t-iterative”, “half-width” and “ratio method”. Most of them are based on confidence intervals over the mean value of the variables, however the difference among them arise in the considerations done when developing the method. After the implementation, the t-iterative gives us exact values regarding the required simulations number, however it needs to iterate and compute all the values after each simulation run and this could be a problem when the number of sim runs rises. The other methods does not provide an exact value but an approximation of the final required number of sim runs based on a sample defined by the user. Regarding the sample, the higher, the higher the accurate of the method. When the validation was performed, only the t-iterative and the half-width method were considered as valid. The values given by the half-width are considered valid since they are generally higher than the values where the t-iterative method stops. The problem for this method is that depends highly on the first simulation values and the range of them.

To end up, an experiment to rate the performance of the two valid methods was conducted for different confidence levels. In the Figure 2, the required number to achieve the level of confidence of each variable is depicted.

.

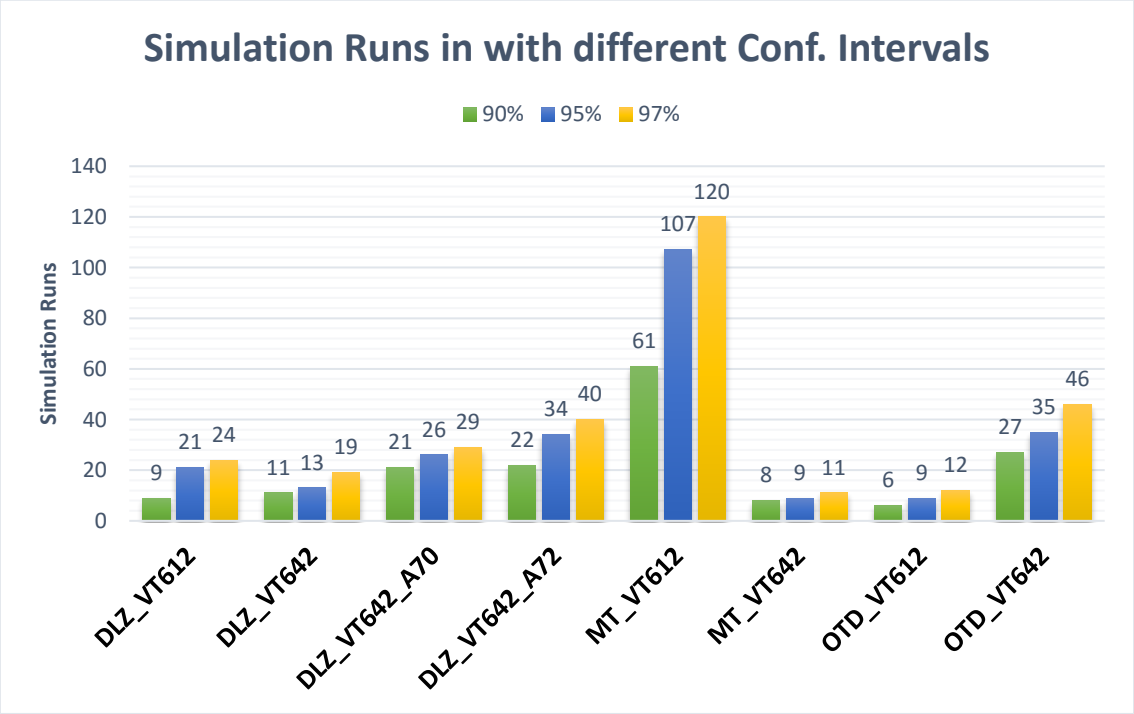


Figure 2

4. Outlook

Given the clear difference between the variable MT_VT612 or Mean tardiness of the train model VT612 and the others, we propose to conduct simulations to that reach only the limits of the next critical variable. In the example presented is OTD_VT642. With 46 simulation runs we could achieve a 97% confidence level on the average in all variables but one. Another option could be rising the level of error that we can't to assume in the variable MT_612, this would lead us to lower requirements.

Regarding the other implemented method (half-width) we conclude that it is useful if what the user looks for is not an exact value but an approximation to it .It must be used carefully when the user makes assumptions based on this results. Finally with the objective of having the best from both method they have been simultaneously implemented in such a way that after reaching the initial number of simulation runs we can check in the software an approximation of the number of iterations required and then act in consequence.

5. Bibliography

[Bake09] Baker K.R. und Trietsch D. (2009): *Principles of Sequencing and Scheduling*.

[Bang10] Bangsow S. (2010): *Manufacturing Simulation with Plant Simulation and SimTalk*. Springer.

[Car14] Carolin Kellenbrink, Felix Herde, Steffen C. Eickemeyer, Thorben Kuprat, Peter Nyhuis (2014): Planning the regeneration processes of complex capital goods.

[Den16] Denkena B.; Dittrich M.A. und Georgiadis A. (2016): Combining in-house pooling and sequencing for product regeneration by means of event-driven simulation. In: *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering*..

[Kis06] Kister T. und Hawkins B. (2006): Historical View of Maintenance. In *Maintenance Planning and Scheduling: Streamline Your Organization for a Lean Environment*. S. 1-18.

[Marc10] Resico M.F. (2010): *Introduccion a la Economía Social de Mercado*.

[Ross16] Rossetti M.D. (2016): Analyzing Simulation Output. In Wiley (Hg.), *Simulation Modeling and Arena*. 2nd. Aufl. John Wiley & Sons, Inc. S. 300-384.

[VRS12] V.R.S. Raju, O.P. Gandhi, and S.G. Deshmukh (2012): Maintenance, Repair and Overhaul Performance Indicators for Military Aircrafts. *Defence Science Journal*.

INVESTIGATION OF VALIDATION METHODS FOR THE EVENT-DRIVEN SIMULATION OF MRO PROCESSES

Autor: Ignacio Marchena

Director: Alexander Georgiadis

RESUMEN DEL PROYECTO

1. Introducción

La compañía „“DB Fahrzeuginstandhaltung GmbH” o DB Vehicle Maintenance, tiene desarrollados en el software Tecnomatix: Plant Simulation, diferentes procesos de mantenimiento con la intención de mejorar dichos procesos a través de experimentos o solventar problemas que pudieran presentarse. En este contexto se sitúa la necesidad de validar los resultados de dichas simulaciones, puesto que actualmente la compañía elige un número alto aleatorio y posteriormente analiza los resultados con Excel resultando por tanto en un proceso no automatizado y muchas veces tedioso. El presente trabajo analiza diferentes métodos de cálculo del número de repeticiones necesarias para alcanzar un cierto grado de confianza estadística, más tarde se presenta una metodología para aplicar dos de los métodos analizados así como una implementación en el software de los mismos. Finalmente, los métodos son validados en Excel y se realizan experimentos para valorar el impacto de los mismos y extraer conclusiones.

1.1 Estado del arte

En el estado del arte del proyecto se estudian no solo los métodos relativos a la simulación sino todo el proceso para aportar una visión del conjunto al trabajo. El apartado comienza con descripciones de los tipos de capital que se emplean en industrias dedicadas a MRO que, debido a su complejidad, coste, número de interdependencias y número de partes involucradas justifican la necesidad de realizar un “Planning and control” de las reparaciones de las mismas. En la figura 1, se puede apreciar la diferencia entre procesos de ensamblajes simples y complejos [Bake09] [Kis06] [Marc10] [Den16].

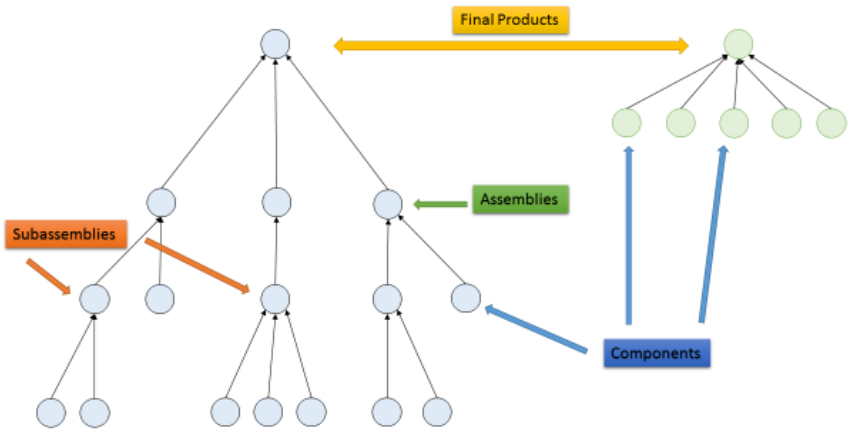


Figura 1

A raíz de las citadas características, se nace la necesidad de las simulaciones. Estas permiten realizar experimentos sin coste añadido o sin parar la los procesos de mantenimiento [Car14] [VRS12]. [Ross16] [Bang10].

1.2 Objeto del Proyecto

El presente proyecto tiene como objetivo la investigación de métodos para validar resultados dados en simulaciones de tipo “Discrete even-driven”. Este tipo de simulaciones se caracterizan porque cada acción sucede en un momento concreto del tiempo y cambia el estado del sistema. Aunque el proceso real podría ser tomando como infinito, las simulaciones realizadas son finitas y es por tanto ese campo de las simulaciones sobre el que se investiga. El siguiente objetivo del proyecto es la implementación de dos métodos en el software Plant Simulation sobre el proceso de reparación de sets de ruedas ya desarrollado por DB.

2. Metodología

Para abordar el problema se dividió en 4 etapas, la primera parte, donde se ha realizado una investigación y resumen de las tecnologías presentes relacionadas con el campo en el que se desarrollaron las simulaciones y otras que podrían ser aplicadas en un futuro con algunos cambios y consideraciones. La segunda parte consta de la metodología para la implementación de los métodos elegidos. En ella se analizan las variables de entrada “inputs” y de salida “outputs ” necesarios para cada uno de los métodos que serán implementados posteriormente asi como los flujos de información en dichos métodos. La implementación es llevada a cabo en una tercera fase y por ultimo se lleva a cabo una validación en Excel asi como un experimento en el que se comparan distintos tipos de simulación para distintos valores de niveles de confianza y error en alguna varaible.

3. Resultados

Los métodos implementados en el software fueron el t-iterative, el half-width method y el ratio method. Los 3 están basados en intervalos de confianza sobre la media, y la diferencia entre ambos radica en la serie de suposiciones que hacen. Tras la implementación de los mismos, se observa que el t-iterative da unos resultados exactos en cuanto al número de simulaciones necesarias. Itera hasta que el grado de confianza deseado es alcanzado. Los otros dos métodos, no dan un valor 100% exacto, sin embargo son útiles ya que permiten establecer una aproximación del número de iteraciones necesarias a partir de una muestra inicial. Al realizar la validación de los métodos se comprobó que solo el t-iterative y el half-width dan resultados válidos. Se considera que un resultado del half-width son válidos ya que en general dan valores mayores que los necesarios sin embargo cabe mencionar que este método tiene una gran dependencia de los valores iniciales que aparezcan en la simulación, y el rango donde estén los mismos.

Por último, se realizó un experimento con los dos métodos considerados validos en los que se prueban para distintas configuraciones de niveles de confianza. En la Figura 2, aparecen el número de simulaciones necesarias para cada variable medida y para cada grado de confianza dados unos valores de error asumidos por la compañía.

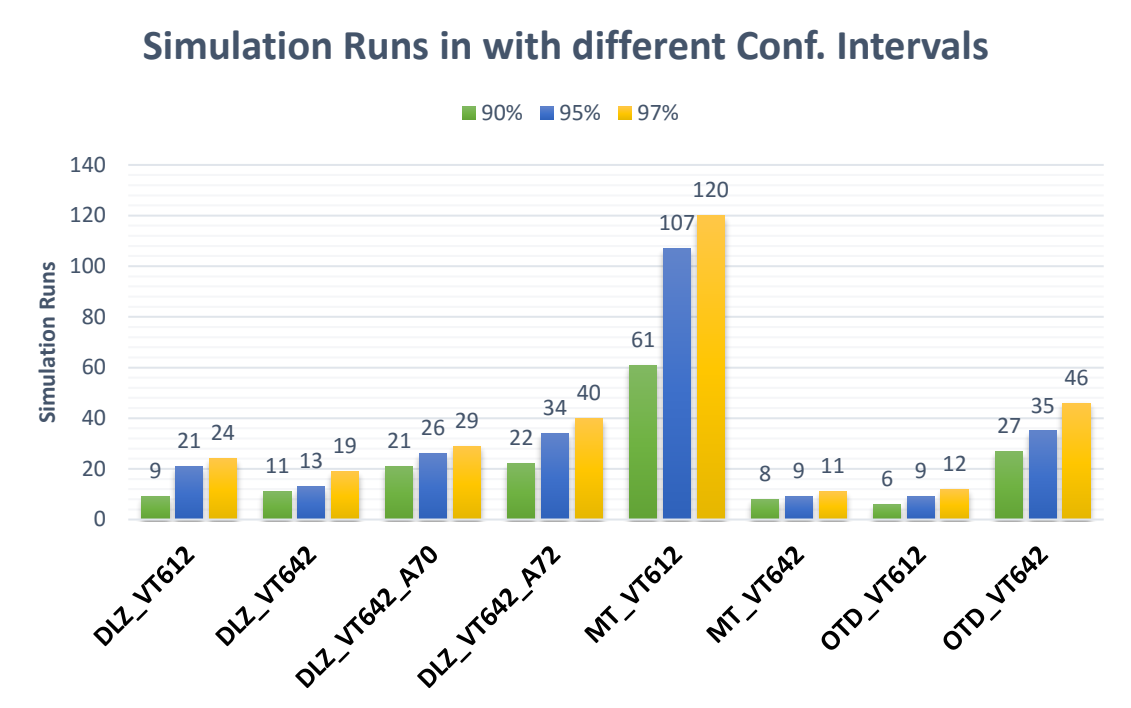


Figura 2

4. Conclusiones

Dada la clara diferencia existente entra la variable MT_VT612, que corresponde al tiempo medio de tardanza del modelo VT612, y el resto de variables analizadas se propone que se analice el modelo para el límite de la segunda variable con mayor número de simulaciones (OTD_VT642). Con este valor, 46 simulaciones, conseguimos una confianza del 97% en todas las variables salvo en la anteriormente mencionada. Otra posible opción sería aumentar el error que se asume en la variable permitiendo así que el deseado grado de confianza se alcance antes.

En relación al otro método válido implementado, half-width method, la conclusión extraída es que es de utilidad si lo que se busca no es tanto una precisión exacta en la validación del resultado como una aproximación del número de simulaciones necesarias. En la última versión de la implementación se acoplan ambos métodos de tal manera que a partir del número inicial de simulaciones fijadas, tenemos una estimación del número necesario gracias al half-width method; este número nos permite decidir en primera instancia cuando puede ser recomendable detener la simulación si la reducción del error en relación a hacer una simulación más es pequeña.

5. Bibliografia

[Bake09] Baker K.R. und Trietsch D. (2009): *Principles of Sequencing and Scheduling*.

[Bang10] Bangsow S. (2010): *Manufacturing Simulation with Plant Simulation and SimTalk*. Springer.

[Car14] Carolin Kellenbrink, Felix Herde, Steffen C. Eickemeyer, Thorben Kuprat, Peter Nyhuis (2014): Planning the regeneration processes of complex capital goods.

[Den16] Denkena B.; Dittrich M.A. und Georgiadis A. (2016): Combining in-house pooling and sequencing for product regeneration by means of event-driven simulation. In: *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering*..

[Kis06] Kister T. und Hawkins B. (2006): Historical View of Maintenance. In *Maintenance Planning and Scheduling: Streamline Your Organization for a Lean Environment*. S. 1-18.

[Marc10] Resico M.F. (2010): *Introduccion a la Economía Social de Mercado*.

[Ross16] Rossetti M.D. (2016): Analyzing Simulation Output. In Wiley (Hg.), *Simulation Modeling and Arena*. 2nd. Aufl. John Wiley & Sons, Inc. S. 300-384.

[VRS12] V.R.S. Raju, O.P. Gandhi, and S.G. Deshmukh (2012): Maintenance, Repair and Overhaul Performance Indicators for Military Aircrafts. *Defence Science Journal*.

INVESTIGATION OF VALIDATION METHODS FOR THE EVENT-DRIVEN SIMULATION OF MRO PROCESSES

Author: Ignacio Marchena

Tutor: Alexander Georgiadis

SUMMARY OF THE THESIS

1. Introduction

The firma „“DB Fahrzeuginstandhaltung GmbH” o DB Vehicle Maintenance, have developed in the software Tecnomatix: Plant Simulation different processes aimed to improve the performance of them through experiments or just to solve problems that could appear eventually. Is in this context where the project find the objective of validate the results of those simulations. In the current state, the researchers of the company choose a high random number of simulation runs and later on analyse the results on excel files. Thus is a process far from being automatic. The present dissertation analyses different methods to compute the necessary number of simulation runs to achieve a certain degree of statistical confidence, later on, a methodology to implement two of those methods as well as the implementation itself in the software is deeply explained. Finally the methods are validated on excel and some experiments are performed in order to value the impact of the changes and bring some conclusions to light.

1.1 State of the art

In the current state of the technology, the methods related to simulation plus the whole process of MRO, are investigated in order to provide a ground for the work. The sections starts with description of the kinds of capital goods used in MRO industries that due to its complexity, investment cost, high number of interdependences and high number of parts involved on them justify the necessity of carry out planning and scheduling stage of the capital good that will be repaired. In the figure 1, we can see the different between simple and complex assembly processes [Bake09] [Kis06] [Marc10] [Den16].

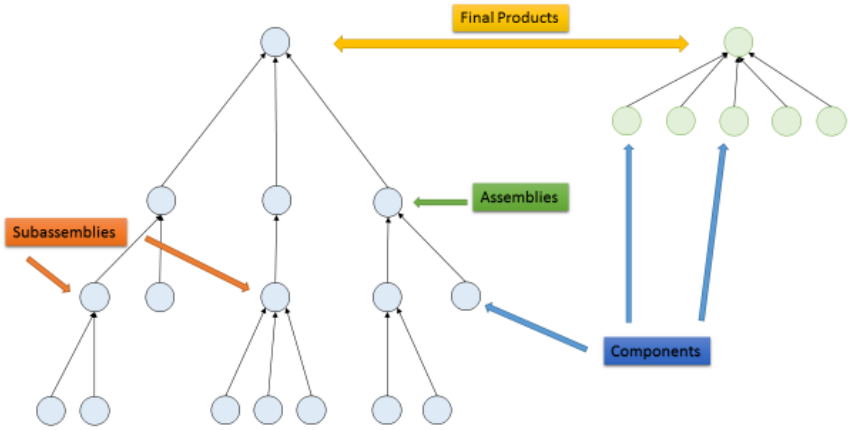


Figure 1

The necessity to carry out simulations arise from the above mentioned characteristics of the goods. The simulations allow the companies to perform experiments without added cost and without stopping the maintenance processes [Car14] [VRS12]. [Ross16] [Bang10].

1.2 Objective of the Project

The work is aimed towards investigation of validation methods for discrete event-driven simulations. In this kind of simulations, each action happen at a specific point in time and change the state of the system. The process faced in the thesis is somehow infinite since new and new parts to repair can come in the future. However, the simulation model considers only one year of repairs, and therefore the methods investigated are focused on finite simulations. The next target is the implementation of two methods in the software Plant Simulation in the smart wheel set maintenance process developed by DB.

2. Methodology

To face the problem, it has been divided into four stages. In the first one, we carry out an investigation and summary of the technologies related with the field on which the simulations take part and others that could be applied in the future under some considerations. The second part contains the methodology for the implementation of the chosen methods. On it, the inputs, outputs and information flows are analysed. The implementation of the methods are developed in a third phase of the project and finally the validation is made in excel files.

3. Results

The methods chosen to be implemented in the software were “t-iterative”, “half-width” and “ratio method”. Most of them are based on confidence intervals over the mean value of the variables, however the difference among them arise in the considerations done when developing the method. After the implementation, the t-iterative gives us exact values regarding the required simulations number, however it needs to iterate and compute all the values after each simulation run and this could be a problem when the number of sim runs rises. The other methods does not provide an exact value but an approximation of the final required number of sim runs based on a sample defined by the user. Regarding the sample, the higher, the higher the accurate of the method. When the validation was performed, only the t-iterative and the half-width method were considered as valid. The values given by the half-width are considered valid since they are generally higher than the values where the t-iterative method stops. The problem for this method is that depends highly on the first simulation values and the range of them.

To end up, an experiment to rate the performance of the two valid methods was conducted for different confidence levels. In the Figure 2, the required number to achieve the level of confidence of each variable is depicted.

.

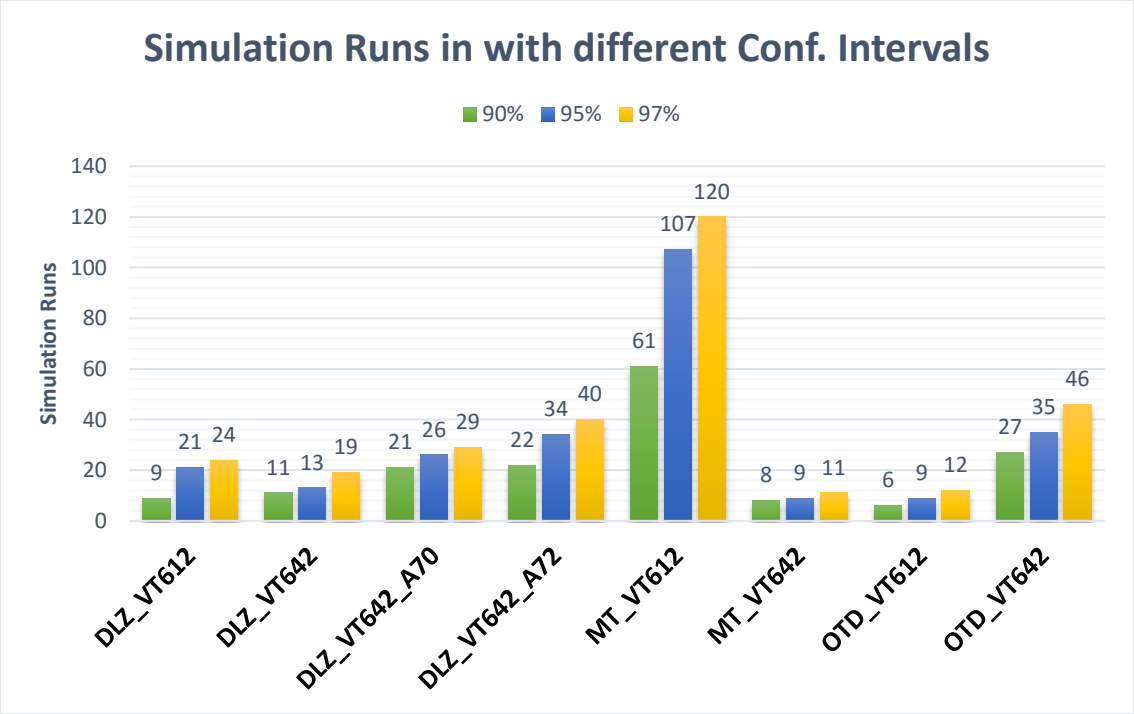


Figure 2

4. Outlook

Given the clear difference between the variable MT_VT612 or Mean tardiness of the train model VT612 and the others, we propose to conduct simulations to that reach only the limits of the next critical variable. In the example presented is OTD_VT642. With 46 simulation runs we could achieve a 97% confidence level on the average in all variables but one. Another option could be rising the level of error that we can't to assume in the variable MT_612, this would lead us to lower requirements.

Regarding the other implemented method (half-width) we conclude that it is useful if what the user looks for is not an exact value but an approximation to it .It must be used carefully when the user makes assumptions based on this results. Finally with the objective of having the best from both method they have been simultaneously implemented in such a way that after reaching the initial number of simulation runs we can check in the software an approximation of the number of iterations required and then act in consequence.

5. Bibliography

[Bake09] Baker K.R. und Trietsch D. (2009): *Principles of Sequencing and Scheduling*.

[Bang10] Bangsow S. (2010): *Manufacturing Simulation with Plant Simulation and SimTalk*. Springer.

[Car14] Carolin Kellenbrink, Felix Herde, Steffen C. Eickemeyer, Thorben Kuprat, Peter Nyhuis (2014): Planning the regeneration processes of complex capital goods.

[Den16] Denkena B.; Dittrich M.A. und Georgiadis A. (2016): Combining in-house pooling and sequencing for product regeneration by means of event-driven simulation. In: *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering*..

[Kis06] Kister T. und Hawkins B. (2006): Historical View of Maintenance. In *Maintenance Planning and Scheduling: Streamline Your Organization for a Lean Environment*. S. 1-18.

[Marc10] Resico M.F. (2010): *Introduccion a la Economía Social de Mercado*.

[Ross16] Rossetti M.D. (2016): Analyzing Simulation Output. In Wiley (Hg.), *Simulation Modeling and Arena*. 2nd. Aufl. John Wiley & Sons, Inc. S. 300-384.

[VRS12] V.R.S. Raju, O.P. Gandhi, and S.G. Deshmukh (2012): Maintenance, Repair and Overhaul Performance Indicators for Military Aircrafts. *Defence Science Journal*.

Acknowledgements

To my family

To my friends

Index of Contents

List of Figures

List of Charts

List of Abbreviations

Symbol	Description	Unit
PSS	Product Service System	[...]
MRO	Repair Maintain and Overhaul	[...]
FMS	Flexible Manufacturing System	[...]
RMS	Reconfigurable Manufacturing Systems	[...]
TPM	Total Productive Maintenance	[...]
LTB	Large Final Order	[...]
MU	Mobile Unit	[…]
SWOT	Strengths, Weaknesses, Opportunities and Threats	[…]

1 INTRODUCTION

In the context of the maintenance, repair and overhaul of complex capital goods the problems that need to be faced such us the uncertain condition of the different components and the interdependencies among them, lead to the necessity of planning and scheduling of the different process. In order to save costs, due to the fact that the “Smart Wheel Set” are expensive, and to reach the optimal solution, the industry is clearly oriented towards the development of simulation models that replicate the behaviour of their systems. The model allows studies without stopping the real process and permits the researcher to experiment and perform a SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis. However, it is easy to find that the outputs from these simulations aren’t validated, and that the number of simulation runs needed is chosen randomly. The thesis is referred to a real case problem: the company “DB Fahrzeuginstandhaltung GmbH″ or DB Vehicle Maintenance, has a simulation model for the “Smart Wheel Set” in the Siemen’s software: Tecnomatix Plant Simulation and the results are not validated. Therefore, in the work, an approach to the problem is analysed in the state of the art, where the MRO processes are deeply explained highlighting the objectives and the problems. Afterwards a description of the strategies of planning and scheduling. Finally, the section ends with an investigation of the different methods available to be implemented in the software. The dissertation proceeds with a description of the software exploited by Deutsche Bahn, here the different elements of the software and specifically those used to implement the method are described in other to give the reader the concepts to understand the implementation. The later chapters presents the strategies followed to implement the validation methods and the code explained down to the last detail in addition of a validation of the method in Excel and an analysis of the performance. At the end, a conclusion has been drawn up and some possibilities for further research are presented.

2 STATE OF THE ART.

2.1 Regeneration of complex capital goods

In the first chapter of the state of the art, description from the sort of goods employ in our regeneration process will be described. Restoring or regenerating processes of this kind of goods has become a main goal for companies since they represent large investment costs.

2.1.1 Complex Capital Goods

In the classic economic, three types of goods, known as factors of production or inputs are described in order to produce others goods and services (outputs), this inputs (land, labour and capital) are described below in order to give us a proper approach to the kind of goods that will be handle in the project. The first input, land, is fundamental to the production of all kind of good. Lands include all natural resources (such as mineral deposits, forest or fish stocks) .The second, described from an economic perspective is labour or human capital, it is a measure of the work done by human beings. Some theories are nowadays relating this concept not with the actual work but with the skills that the workers may possess. Finally, the third one, capital goods are assets with a useful life of more than one year that are used for the production of incomes.

Capital goods and capital assets refers usually to the same concept however is the context which determines which one is used. Capital goods are use in economic analysis and capital assets is a term mostly used in accounting and finance. A capital good is any type of asset that is used to produce other goods or helps in performance services that a company sells to others. This includes buildings, factories, vehicles, furniture, computers or heavy machinery. Capital goods are therefore already produced durable goods or any non-financial asset that is used in production of goods and services [Men07].

This previous description can give us an idea of what kind of goods and what kind of related problems will be solved in this project; nevertheless we also should differ them from consumer goods. Consumer goods are outputs, final goods or products that are

purchased for consumption by the average consumer, they are the result of a production and manufacturing process. Some examples of consumer goods are clothing, food or jewellery.

On the other hand, we could define complex capital goods as these heavy equipment (such as excavators, forklifts, generators, metal-forming or metal-working machines, vehicles, wind turbines diesel locomotives or aircraft engines) which require a relatively large investment, and are bought to be used over several years. This huge investment made by the companies is therefore its main reason to control the condition of the goods as well as to maximize its utility rate by increasing their lifetime cycle. By an effective maintenance significant values can be realized when their lives are prolonged. [Ste13]

Capital goods have played an important role in capital formation and industrialization process in the leading industrial countries in 19th century and for those countries which started to industrialize after World War II, such as Japan and Soviet Union [Bru87].Rosenberg [Ros03], noted that the capital goods sector has been playing a crucial role in the process of technological innovation.

Welp [Wel99] stated that the product structure fix the sort, number and interrelationships between parts and assemblies. The complexity of the product is highly influenced by the number of levels of assembly and the number of items among the product and assemblies. This kind of products are highly customized with multiple levels of product structure. The product structure of a simple and complex process is shown in Figure 2-1.

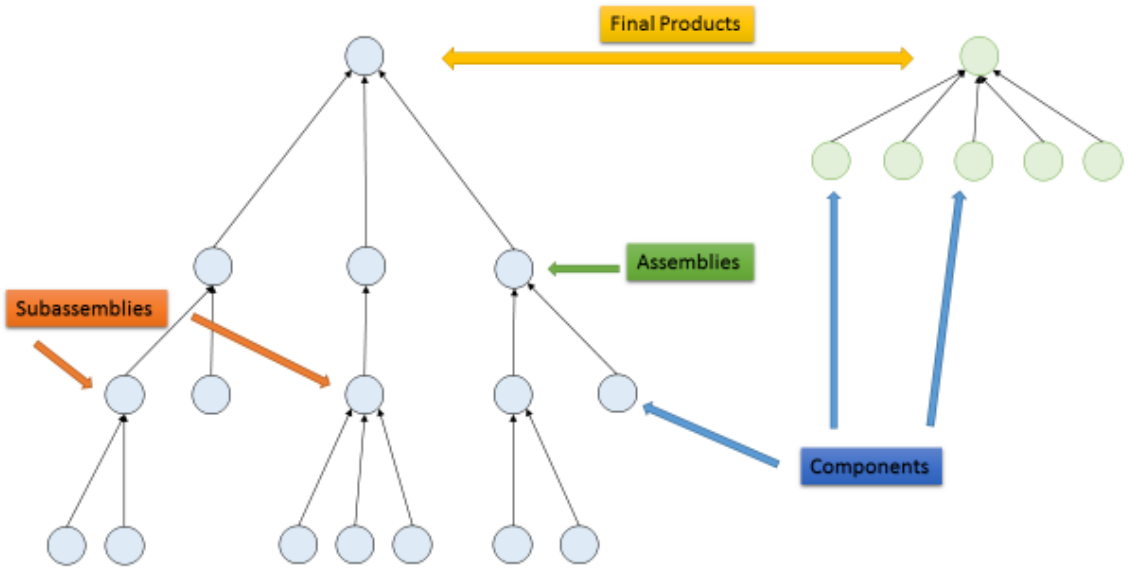


Figure 2-1 Complex vs Simple assembly process

Complex capital goods are characterized by huge number of different components and substantial interdependencies among those components; it appear also as a common characteristic that these products have complex assemblies with many stages of manufacture and assembly. This components need to be substituted or restored after their life cycle since these part's condition gets worse during their lifetime and have high residual value [Lau01].

In addition to this inherent characteristic, it also important to keep in mind some other issues born from the product complexity itself. Due date assignment is an important part of production planning since it specifies the completion time of the product.

When designing the production plan, one of the most important things to take into account is the uncertainty in the manufacturing system caused by processing time variability, material supply changes, machine failures and repairs, customer demand changes, human errors and absenteeism. The manufacture and therefore the regeneration process is characterized by huge levels of uncertainty, unanticipated demand, customized specification and long lead-times [Xie11].

2.1.2 Regeneration Processes (MRO)

In the greater scheme of things, the practice maintenance has not been around very long if we compare to the creation of the first tools. At the beginning, the maintenance mode was “run to failure” due to there were only stone tools. However, around 9000 BC when the stone tools were re-flaked as the edges dulled or broken handles were repaired could be considered the first appearance of the concept of “repair and overhaul” and therefore an historical moment in maintenance.

The preceding paragraph show us a little bit of history without much to do with the history of modern maintenance, nevertheless is the best way of realizing that maintenance had great importance since the beginning. The Industrial Revolution, as it is referred today, is generally acknowledge as beginning around 1750 and lasting until the First World War. The concept of industrial revolution is a convenience to designate a period in which sufficient innovations taken part. At, and just after, the turn of the twentieth century, two events changed the way in which maintenance is performed. The first of these was Henry Ford’s creation of the modern assembly line and the second was First World War.

From 1908 to 1913 the assembly line from Ford’s Model T increase its production from 728 min/car to 93 minutes to adjust to the growing demand. To maintain rates around 1000 cars/day assembly line stoppages could not be tolerated and therefore its primary method to ensure the smooth and consistent flow of each chassis through the assembly line was “planned maintenance” [Kis06].

The second event that highly influence maintenance operations was the First World War. The evolving role of aircraft during the war lead to more hours in the air and therefore rise the number of equipment failures which by that time were most often fatal. Pilots, who were the first interested into returning from each mission, developed pre-mission checklist for determining a plane’s combat airworthiness. After the war, the checklists became through and comprehensive documents for identifying required maintenance and repairs based on the experience of hundreds of pilots. Another example of first maintenance planning can be found on the first tank, developed by the British army. Its maintenance became an important issue and some procedures are known such as engine oil replaced on a time-based schedule or weaponry was completely torn down, cleaned and reassembled daily [Kis06].

With the Second World War, manufacturing manpower shortages, increasing demand on war products and consumer goods lead into another incremental impact on maintenance. Toyota Automobile Group, from the Toyoda family, was forced to scrap their previous plant operations and reorganize because of their support to Japanese war machinery.

They improve existing processes to better support assembly operations. Toyota took two concepts from the U.S.; the assembly line production system and the supermarket concept provided a basis of a continuous supply of materials just as the supermarket provides a continuous supply of goods to the consumer. By developing these concepts Around 30 years later Toyota's production system would be known as "Lean Manufacturing". On Figure 2-2 the evolution of different manufacturing systems through the history is shown.

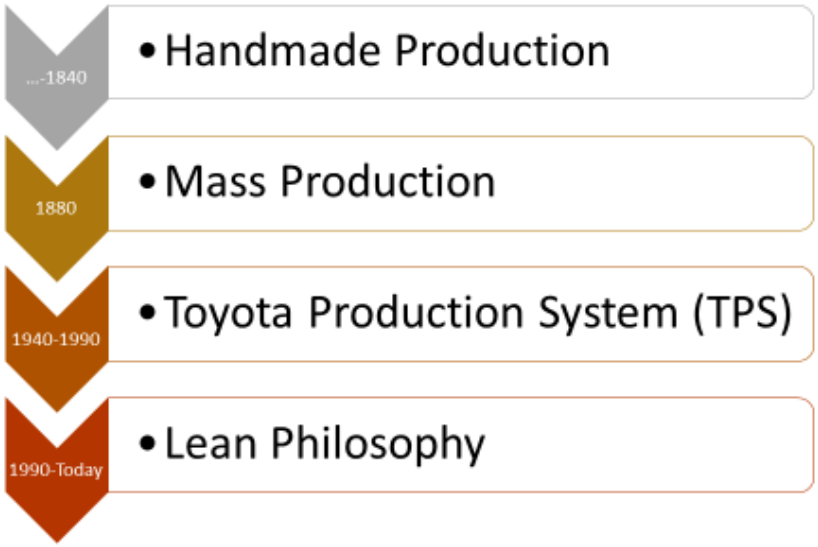


Figure 2-2 History of Manufacturing Development

To sum up, regeneration processes have been involved in the industry since the first tools until the complex maintenance processes that are hold on our days. Their importance have risen along the years, this fact can be deduced since *"the real advancements in the art of maintenance have been realized during the last 240 years and the 95% of these have occurred in the last 25-35 years"* [Kis06]. It is important however to describe what are the goals of a maintenance process in general and the differences when the process is applied to complex capital goods.

As a consequence of the increasing challenges, manufacturing enterprises are making efforts to provide high value added services to their traditional design and manufacture. In this context born the concept of Product Service System (PSS). The idea is to exploit the potential benefits of integrating product development with related services. In [Zhu12] PSS is described as a "*a system of products, services, networks of 'players' and supporting infrastructure that continuously strives to be competitive, satisfy customer needs and have a lower environmental impact than traditional business models".* However, PSS is a business strategy that integrate all the agents throughout the entire product life cycle and can be divided into some categories such as those found in the work of Cook [Cook06] and Morelli [More03]. Especially Morelli [More03] describes the "Product-Oriented PSS" which will be our business environment:

- Product-oriented PSS: selling the product in a traditional manner, with the additional services such as after-sales service, repair, maintenance, reuse and recycling [More03].

Maintenance can be described as a process in charge of ensure that the system makes its intended functions during its life cycle [Zhu12].Other authors, like E.Viles [Vil07] define the goal of the maintenance beyond the repair itself and state that is a way of improving reliability as well as to gain relevant information. In order to provide a concrete definition of the MRO processes the three terms are three different processes that are now described:

❖ Maintenance

Preventive in nature; in maintenance, *"activities are carried out as per servicing schedules at stipulated periodicities […]. After this maintenance activities, it is expected that the system is serviceable und available till the next scheduled maintenance of similar type"* [VRS12]

❖ Repair

Corrective in nature; in repair fault diagnose is considered based on monitored data, symptoms or observations made by the operators. The faulty component is repaired or replaced and then the system is checked for serviceability. If the system fails before the next fixed maintenance, then the reasons must be established. [VRS12]

❖ Overhaul

Includes detailed examination of all component and subsystem in an industrial-type facility and is a combination of preventive, corrective and predictive maintenance. Disassembly to make extensive renovations or revisions. Some factors to determine the performance of an overhaul are the time taking to complete the service, the extent to which the systems are restored to as good as new state or the guaranteed period of failure free operation. [VRS12].

MRO processes must be able to deal with the requirements of inspecting, servicing, repairing, and overhauling production systems. The general objective of MRO is to achieve a high level of reliability and availability as well as a high degree of economic, ecological and social sustainability of a production system [Bier16].

2.1.3 Regeneration of Complex Capital Goods

Regeneration processes of Complex Capital Goods are different from common manufacture of Complex Capital Goods, from repair of simple capital goods or even from remanufacturing processes since MRO processes deal with a combination of the problems shown in all of them.

The main problems faced in the planning of regeneration of complex capital goods is the uncertainty condition of the goods, the large amount of spare parts with interdependencies mixed with different wear levels at the regeneration time and the different repair paths among the different business models lead us to the challenge of planning the process. MRO of complex capital goods face challenges that differ from common production processes. On a common production process the company fixes the production levels and carry out a long term planning that allows it to reduce costs and improve the efficiency of the process. The determination of each other workload becomes a key factor when classifying the processes. The workload, in regeneration of complex capital goods, remains unknown until the diagnoses at the regeneration-service-provider's site [Kelle14]. In addition the regeneration also differs from remanufacturing where "*the re-use of low-value goods is investigated"* [Kelle14].

In the case of regeneration of complex capital goods, we can find different ways to accomplish a reasonable regeneration approach depending on the good's condition and the customers' specific business model that is why the requirements of

regeneration planning and the selection of the most efficient way to made it differ according to customer's business model [Kelle14]. The objectives of the planning of regeneration processes are:

- Recover functional characteristics of the different spare parts
- Guarantee customer satisfaction
- On-time finishing the orders

In order to fulfil the objectives mentioned before, a planning of the regeneration process must be performed in advance. This planning is mostly done by using customized software tools to support regeneration management. Almost all of these tools are self-developed software applications based on Microsoft Excel and/or Access and thereby we can assume that there is no practical relevant, rule based and therefore a method or software suitable for planning or controlling the regeneration of complex capital goods [Kelle14].

In this way, the uncertainty becomes a key factor since it prevent us to make a perfect planning as it would be done in the case of a manufacturing process. In a regeneration process we cannot plan because we don't know the level of wear of our components and therefore we cannot claim if we will need a spare part or not, or even the time that repair process will take.

The main challenge is to carry out a planning after the disassembly and diagnose process, this planning should be able to minimize the cost and allow on-time finishing [Plac14]. In the Figure 2-3, the first steps of the regeneration process are depicted.

Figure 2-3 First steps of a MRO process

2.2 Regeneration Planning and Control

According to Graves [Gra99], regeneration, planning and control entails the acquisition (if necessary) and allocation of limited resources to production activities so as to satisfy customer demand over a specified time horizon. Planning and control are therefore, optimization problems where the objective is to design a plan that meeting demand minimize cost and maximize profit. The underlying optimization problems will depend on the business. Typical decisions are workforce level, production lot sizes, assignment of overtime and sequencing maintenance runs.

In our case, since we have to deal with complex capital goods and due to the uncertainty of these components until they have been disassembled and checked an approach of reconfigurable manufacturing systems (RMS) and flexible manufacturing systems (FMS) is given:

Reconfigurable manufacturing systems (RMS) can be described as highly dynamic and evolving systems designed to cope with unpredictable situations [Hees15]; in addition, ElMaraghy [ElM06] establish a difference between soft and hard reconfiguration. He describes soft reconfiguration as the one fulfilled only with software adaptations or organizational aspects (e.g. additional shifts) since hard reconfiguration requires changes and modifications of hardware. The main characteristic for RMS is their adaptability and therefore its functionality and capacity can be configured as needed to respond to market changes.

In contrast, flexible manufacturing systems (FMS) are built up with all possible set-ups concerning flexibility and functionality and can adapt only within their a priori predetermined flexibility [ElM06].In other words, they are able to process only a predefined spectrum of components. Further adaptations will require substantial efforts. FMS are limited in capacity as well as in functionality but adaptations inside their functionality can be managed rapidly and at low cost, however, they require a huge initial investment.

The regeneration planning and control supports maintenance companies in their order processing, it fulfil the task to plan and control the regeneration process in term of quality and scheduling. [ElM06]. The first requisite for regeneration planning is the formal description of resources and production orders. Thus, there is a need for generic data models representing configuration-based skills of resources and requirements of

maintenance orders [ElM06]. These data model include all the necessary information to approach the maintenance planning. Based on the data models, the configuration management has to select and adapt the different possible configuration of manufacturing resources as well as matching skills of the resources and the order requirements.

As shown in Figure 2-4, with the assistance of data models and configuration management, a suitable sequential resource planning approach is developed.

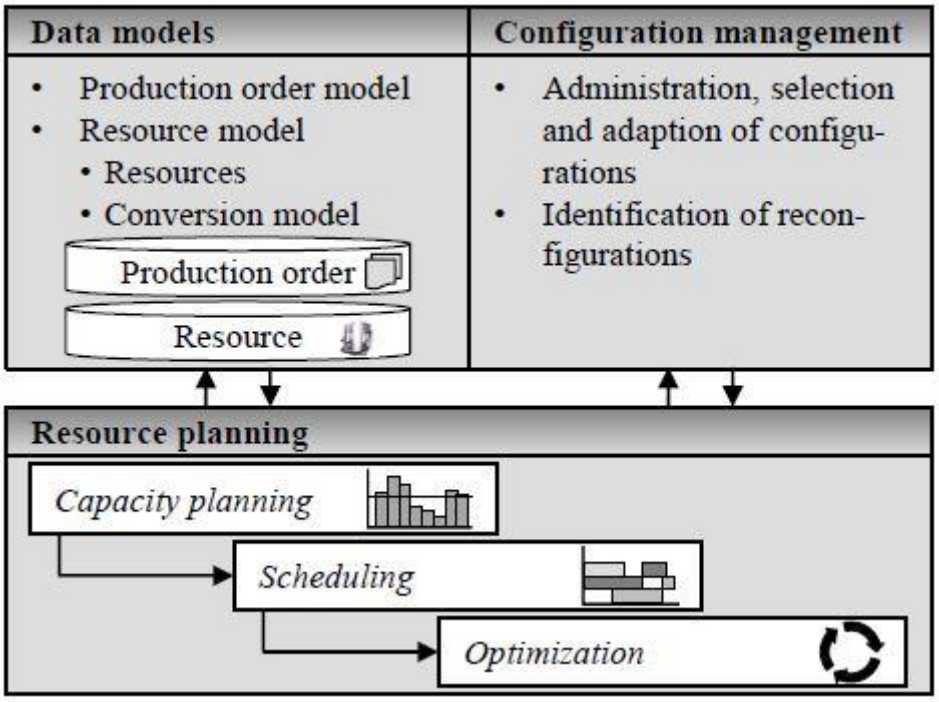


Figure 2-4 Overview and structure of the planning system [Hees15]

The objectives that the maintenance planning try to fulfil are [Kist06]:

1. *"Optimal support of the operational production plan by improving maintenance in the broadest sense, considering both the technical aspects and service provided to the internal customer"*

2. *"Completion of maintenance work when it is needed, in a safe and efficient manner, at the most effective (optimal) cost"*

3. *"Minimization of lost production time due to maintenance"*

4. *“Optimized utilization of maintenance labour and materials through effectively planned and balanced schedules”*

5. *“Equitable resource allocation based on understood criteria and the varying business needs of the internal customers supported”*

6. *“Minimization of labour delay and idle time through effective coordination of all participating functions”*

- **Operating and Maintenance Strategies**

The sort of maintenance philosophy applied during the machine cycle-life is a basic decision that should be properly and carefully taken. This decision should be never conducted by the manufacturer but by operator in order to fit their specific application. Three possible maintenance strategies can be developed:

1) Reactive Maintenance Strategy

It could be described as “wait until something fails”, this strategy should be avoided on all machinery in critical applications. Obviously this strategy has the lowest maintenance cost but can lead to unexpected failures and then major losses [Lau01].

2) Preventive or Aggressive Maintenance Strategy

This kind of maintenance operate with a regular plan that is set up only if proofs of equipment deterioration are founded. Strategies of predictive maintenance favours minimizing cost over maximizing use. The annual cost of for this strategies is only 1-2% of the investment price. The necessity of monitoring the activities or the goods is subtracted by its own definition; however sometimes the moment of setting up the maintenance plan relies on the probability of failure in a specific interval. The benefits are the reduced probability of breakdown and making the equipment life longer, however the main disadvantage is the obligation to stop activities at fixed intervals to perform the checking of the different components [Lau01].

3) Predictive or Proactive Maintenance Strategy

The objective here is to maximize safety against unexpected failures. The idea is to predict the failure and replace or repair the part before it fails. The annual cost, are obviously higher that in predictive maintenance (8-10% of the prime equipment price) because of the higher control levels and warehouse costs. Figure 2-5 does not show the cost of implementing the maintenance strategy but the cost may have in the future as a function of the effectiveness in a general repair process, (e.g.) using a reactive maintenance on aircraft turbine may lead to the necessity to buy a new turbine due to its complete failure instead of just repair some wear parts.

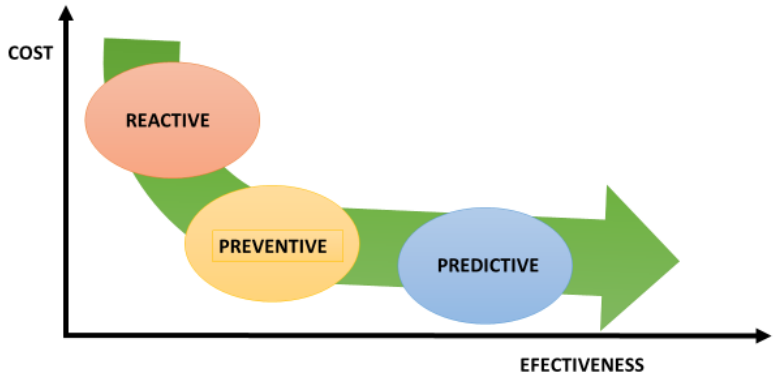


Figure 2-5 Maintenance Strategies

2.2.1 Inventory Management in Regeneration processes

Inventory management permeates decision-making in countless firms and has been extensively studied in the academic and corporate spheres. The key questions, which inventory management seeks to answer are [Wan14]:

- When to order.
- How much to order.
- How much stock to keep as safety stock.

Inventory management includes a bunch of decisions that according to Wanke (2014) [Wan14] *"aim at matching existing demand with the supply of products and materials over space and time in order to achieve specified cost and service level objectives, observing product, operation, and demand characteristics".* This circumstances that should be taken into account for a proper selection of the inventory management

strategy have contributed to the amount of research and articles on different schemes and possibilities known as well as "*classification approaches-aimed at supporting decision making*" [Wan14].

In a railway company, high availability of the trains is basic and as a consequence, broken components must be replaced by good condition components during line maintenance. After this, separately, the defective component is repaired, this allows the operators to reduce the duration of line maintenance.

To enable efficient planning and execution, railway companies need that repair shop achieve a short turnaround time. In this case, timely availability of the resources needed for component repairs is key. To assure the availability of spare parts is in the case of complex capital goods a huge challenge to face due to the components may consists in hundreds of different parts or subcomponents and until disassembly, the number of them that may need replacement remains unknown. From the point of view of the customers, they are not only buying the good at an affordable price but, looking for a balance between the resulting total cost and the availability of the system during its lifetime and regarding to this, sometimes, the support cost for a system through its lifetime is a large part of the total cost of ownership [Behf15].

To achieve a high uptime, capital goods are often repaired by replacing failed parts by ready-to-use parts from inventory. Therefore, providers should offer high availability of the spare parts since only the inspections will reveal which parts are needed in each repair. Besides, demand for each spare part is unpredictable so repair shops have to keep large spare parts safety stocks. Stochastic inventory control methods for safety stock optimization typically set availability targets based on price, lead time and demand volume [Wil15].

Sometimes, due to technological development or new systems introduction, the demand of the spare parts drop causing the manufacturer to decide that is no longer profitable to produce them. If this happen before the service obligations, the company must find a solution. Seeing that, maintenance service provider usually decide to cover future demands considering the uncertainty; this strategy is called Large Final Order or LTB. However the LTB order quantity is very large to attain a high service level which also yields high obsolescence levels at the end of the service period [Behf15].

Due to this reason, companies consider others sourcing options such those shown in [Behf15].

I. *"Repair of failed parts that are returned from the field"*
II. *"Strip phased-out systems for reusable spare parts"*
III. *"Buy second-hand parts on the open market"*
IV. *"Substitute by a compatible part*
V. *"System redesign avoiding the need of the specific spare part*."

Spare parts inventories exist to serve maintenance planning. An excess of spare parts leads to high holding cost and in contrast the lack of the spare parts may result in cost delays with a negative impact for the company.

Since the railway industry involves a large number of parts and some of them are quite expensive, it is important to find an appropriate inventory model to achieve a right balance. Traditionally, spare parts are generally classified according to [Jing15] into four groups: rotables, repairables, expendables and consumables, which are listed in Table 2-1

Rotables	Complex components Normally unlimited number of repairs Normally no scrap is expected Exchange during maintenance
Repairables	Components that can be technical and economically repaired Limited number of repairs and chance to scrap
Expendables	Cannot be repaired and will be scrapped after removal Not economical to be repaired 100% replacement items Standard parts
Consumables	Materials used only once Raw material Chemical material Items that merge on production with new product

Table 2-1 Definition of spare parts groups [Jing15]

Different replenishment strategies are used relying on the different categories of spare parts. In particular, rotables and repairables are mainly based on predicted failures and the planning parameters are finally a management decision. For expendables and consumables, the reorder point system (ROP) is used. In this strategy the input comes from historical demand with estimated variability. Nevertheless, this strategy of inventory management is subjective and imprecise and therefore quite far from been ideal [Jing15]. In a recent survey conducted by Ghobbar [Gho02], 152 out of 175 survey respondents were using the ROP system and about the half were dissatisfied and considering implementing new strategies.

In order to reduce the uncertainty in the demand and the operational costs and increase profit, many companies choose the strategy of inventory pooling since is a case extensively studied in operations management and its benefits are well known. The strategy, that consist on using a common pool of inventory to satisfy two or more sources of random demand has been studied in the context of many operational situations [Alp13]. Some literatures explore how pooling acts as a container of the "operational costs in a high product variety, mitigating supply chain disruptions or striking the right trade-off between operational benefits and fixed costs of product-process flexibility in supply chains" [Alp13]. Other definition provided by Wong [Won07], states the pooling as an arrangement in which different companies share their inventories in order to reduce their excessive inventory cost of their equipment-intensive industry.

This practice can be analysed from two related perspectives: location and product pooling. *"Location pooling refers to the practice of pooling demands from separate geographic markets (e.g., combining the inventory from stores in two different physical locations) […]. Product pooling, on the other hand, refers to the practice of meeting demand for multiple distinct products with a single, "universal" product capable of satisfying the needs of all customers"* [Swi12].

2.2.2 Method for In-house Pooling Inventory

Once that the use of simulations have been properly justified, one of the problems faced is to know how many simulations should we run in our model. In the method for train couplings developed by IFW they carry out a sensitive analysis after combining in-house pooling and sequencing by means of event driven simulation.

The idea of in-house inventory pooling according to [Den16] is transfer the idea of external pooling to the pooling of components in the regeneration place. On external product pooling the inputs that need to be regenerated are processed and after that send to external warehouses. Then, the pool provide the restored components directly to the assembly. The problems that arise from this strategy are cost of inventory purchases and transportation cost. To transfer this idea to an in-house situation, the components that bring higher delays and therefore produce bottlenecks on the repair process are identified by simulations. Some examples of the kind of distributions that can be founded are depicted on Figure 2-6, as it is shown, the distribution depends on the type of component. Once that the components are identified a pool of those is created. In the process, when one of these components needs to be repaired, an already repaired component goes directly to the assembly reducing therefore the cycle time and improving the on-time delivery [Den16].

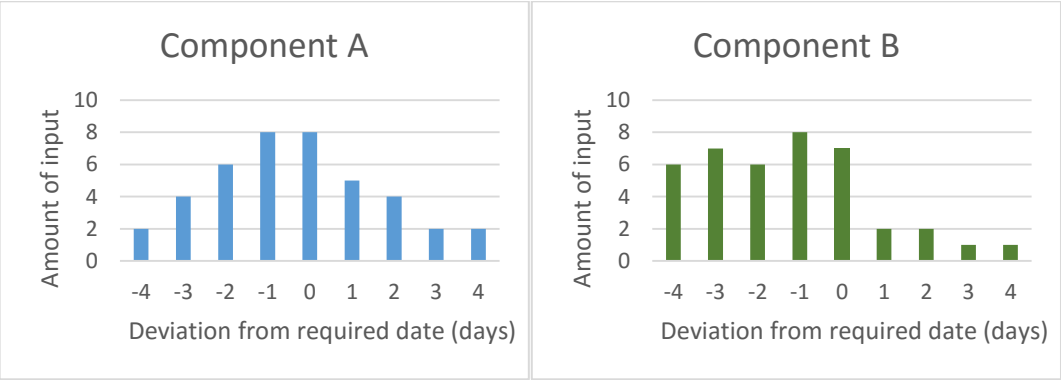


Figure 2-6 Distributions of output delay

The coefficient that evaluate the degree of delay of each component is shown by the equation (1)

$$CI_{ij} = \sum_{t=1}^{Tmax} \frac{t}{T_{max}} \cdot input_{i,j,t} \quad (1)$$

After calculating the component indicator, the algorithm determine the optimal number of pool inventory for the component with the maximum CI through iterative calculations where after each iteration the process is simulated with the new settings and then the cost and benefits are evaluated. If the cost-benefit analysis satisfy some requirements, the iteration stops. If not, then the system increases the pool inventory of the studied component by one unit and closes the loop.

2.2.3 Sequencing & Scheduling

Scheduling is a term used in our everyday vocabulary, however, the definition sometimes is not accurate with what we thing. An accurate definition of scheduling is given in Principles of Sequencing and Scheduling [Bake09]; „*A schedule is a tangible plan or document [...], that usually tell us when things are supposed to happen. It shows us a plan for the timing of certain activities and answers the question: If all goes well, when will a particular event take place?* “. The answer of „when “is usually presented with timing information, however, it can also be equally useful in terms of sequence rather than timing. On a process when we face a problem we design the sequence on the first step and decide how to select the next task. The start time is planned on a second step and perhaps the completion time of each task. Also the determination of safety time is part of the second step. Scheduling problems in industry often contain a set of task to be done and a set of resources to be allocated to perform these task.

A specific approach for a regeneration management is required in order to guarantee an efficient planning process and in order to cope with the regeneration specific characteristics. First, it is necessary to choose a method that ensures a reliable capacity planning. Second, based on capacity planning and the resulting information regarding used and unused capacities, a decision to accept or deny further regeneration orders must be made. If an order is accepted, more specific scheduling must be undertaken, and the most efficient regeneration mode should be determined.

To further improve regeneration-planning processes, load adjustments and pool-management strategies should be considered. When planning capacities and regeneration modes and when deciding whether to accept further orders, customers' preferences concerning regeneration modes must be considered. In order to cope the first step of the regeneration process, the company should decide which risk want to assume.

In order to face the issues of production control arising from MRO specific characteristic, many experts suggest the use of sequencing rules. These rules provide the possibility to reduce waiting times and therefore they are used to determine the priority of each job or component in production whereas jobs with the highest priority at queues should be processes first [Geo14]. Scheduling rules fix which waiting job or component should be scheduled in preference to the others. The five well-known sequencing rules are depicted in Table 2-2.The Fix-and-Continue algorithm developed by the Institute of Production Engineering and Machine Tools (IFW)

Rule	Full Rule Name	Description
FIFO	First-In, First-Out	The job or component which arrives firs is processed first
SPT	Shortest Processing Time	The job or component with the longest operation processing time is processed first
LPT	Longest Processing Time	The job or component with the longest operation processing time is processed first
EDD	Earliest Due Date	The job or component with the earliest due date is processed first
JST	Job Slack Time	The job or component with the minimum slack is processed first

Table 2-2 Common Sequencing Rules [Geo14]

In order to solve the problems like reducing the unscheduled waiting times the Fix-and-Continue algorithm was developed by the Institute of Production Engineering and Machine Tools (IFW). First this algorithm determine through simulations the component with the highest buffer idle ratio measured and weight that component with

the highest priority. Since this component is prioritized, its idle buffer ration will decrease and increase the ratio for the other components. Then the simulation is rerun with the new settings in order to find the next component with the highest priority [Geo14].

- **Scheduling Dilemma**

To improve and differentiate from others and since by way product features in no longer enough for marketing success in a company, delivery reliability and delivery time have become as important in buying criteria as product quality or price. The conflict found in the production logistic known as the scheduling dilemma involves the different objectives, short throughput time, high schedule reliability, low work in process (WIP) level and high machine utilization since they have adverse orientations. In figure the dilemma explained before is presented [Nyhu06].

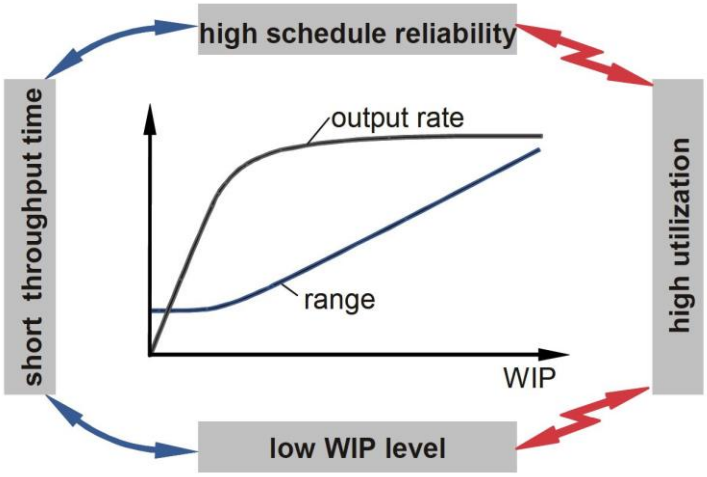


Figure 2-7 Scheduling Dilemma [Nyhu06]

What Figure 2-7 represents is that if for example we decide to focus on high utilization of work systems, it calls for high WIP level in order to prevent possible interruptions in the material flow. However, high WIP levels lead to long throughput times. Thus high machine utilization and short throughput times cannot be achieved. In order to overcome this dilemma, it is essential to map the relationships between the effects of different production logistics performance measures as well as to describe the behaviour of the logistic system [Nyhu06].

Popular methods for modelling the behaviour of the logistics system in individual work systems or production areas are the Queuing Theory, simulation, and the Theory of Logistic Operating Curves developed at the IFA. These three methods can be compared by using the criteria applications, cost of modelling, level of detail, quality of results and applicability in practice. Queuing Theory and simulation are often not entirely suitable for modelling logistic relationships, especially in the mapping of real production contexts [Nyhu05].

"The theory of logistic operating curves is based on a deductive-experimental effects model which was used as the starting point for developing approximation equations for the mathematical description of the real Logistic Operating Curves" [Nyhu06].These curves try to reduce the complexity and the cost of evaluating the behaviour of the logistic system. They implement a way of positioning a work system of a production area in the conflict between the logistic performance measures [Nyhu06].

In order to depict the behaviour of the process some of these logistic operating curves will be explained. First of all the Assembly Throughput Diagram plots the cumulatively over their dates of the entry of component or finishing of the assembly order. In the example shown in Figure 2-8, planned dates of entry components and orders completion, as well as the completion curve are also shown to provide more information [Beck11].

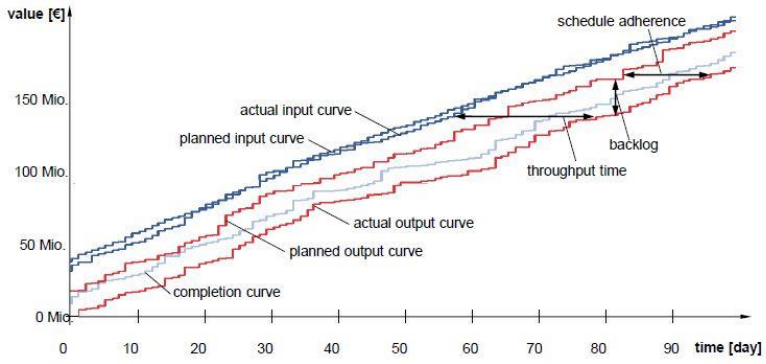


Figure 2-8 Assembly Throughput Diagram [Beck11]

The beginning of the input curve is determined by the WIP (work in progress) found on the workstation. Some other features can be evaluated from this diagram measuring the gaps between curves, these are shown on Table 2-3.

Type of Measure	Curves Involved	Measure
Horizontal difference	Planned and Actual output	Lateness
Horizontal difference	Actual Input and output	Throughput time
Vertical difference	Planned and Actual output	Backlog

Table 2-3 Results from assembly throughput diagram

The throughput diagram is able to describe the dynamic system behaviour qualitatively and chronologically however, it cannot tell us if the input curve contains or not the required parts by schedule. To face this problem the Supply Diagram is presented on Figure 2-9. It provides a clear description of the scheduled situation of an assembly's supply and shows the lateness of multiple orders or components on a specific period of time.

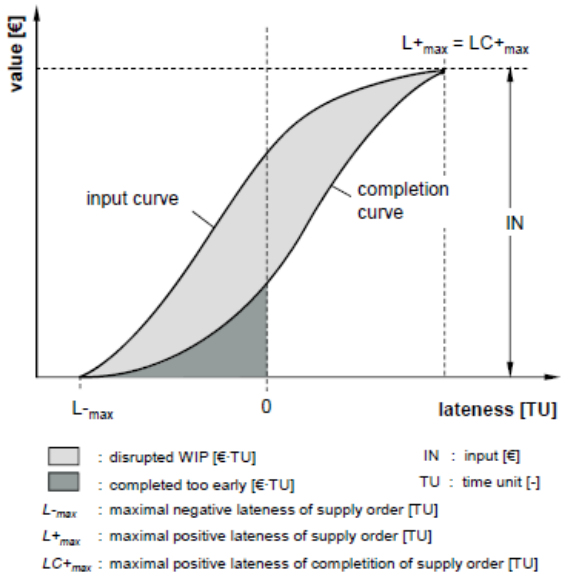


Figure 2-9 Supply Diagram [Beck11]

The diagram shows two different curves; the input, "*represents the input dates of all single components on a chosen period"* [Beck11] sorted by its lateness a weighted with the value. On the other hand the completion curve represents "*the lateness of the last*

order or component which is needed to start the next process in the supply chain", in this case the completion curve is weighted with the value of all the components that are required to finish the order. The area among the two curve is as can be seen in the Figure 2-9 the disrupted WIP or the quantity of WIP waiting before the assembly system [Beck11]. On other words is the value of the already supplied components that cannot yet be processed.

2.3 Simulation

Simulation has been described as „*the process of designing a mathematical or logical model of a real system and the conducting computer-based experiments with the model to describe, explain and predict the behaviour of the real system"* [Wil96]. Scheduling, on the other hand, deal with the problem of deciding the processing times of jobs in a project given the limits of personal, equipment and facilities. However, despite their differences, they can work synergistically toward process improvement. Controlling the operations is an ever-present challenge in operations management in order to maximize economics effectiveness. In the productions processes these challenges usually presents itself in two levels: The long term work balancing or macro level and the daily facility control or micro level [Wil96]. However given the uncertain condition of the goods in the MRO process the challenges will be slightly different.

Optimization through simulation has received considerable attention on recent years by practitioners and researchers. A lot of problems related with optimization in real life like manufacturing systems, supply chain systems, financial management, transportation systems or communication networks are too complex to be modelled analytically. The reason due to this systems cannot be analytically modelled is the different difficulties inherent in the systems such as „*high variability, underlying non-linear dynamics, large problem size and possible multiple objectives"* [Lee08]. Those problems are usually faced with discrete event simulation. A simulation based optimization can be defined as the coupling of an optimization method with simulation in order to test many parameters that maximize performance of the simulated system [Lee08].

When dealing with capital equipment such as in vehicle maintenance, the maintenance service is made usually by following a replacement policy, i.e.; failed component are removed and replaced by a suitable spare part, if the spare part is not available, then the system is unavailable and the repair process is delayed. Later on, the defective component can either be discarded of repaired. If it is repairable, the repair is performed by replacing the defective part by a spare part. A system or prime equipment, may contain hundreds of assemblies and subassemblies, components, parts, organized in multiple indenture levels each of them with multiple options. In some cases each level of maintenance can be carried out in different locations. Real world systems are described as a multi indenture product structure and maintained by a multi echelon repair network [Gha16]. Simulation allows to dynamically determine if a component should be repaired or not upon its failure and given the case on which location should the repair be done.

In comparison with direct experimentation with the real system we can find some reasons that lead us to prefer simulation [Rob64].

- Cost: Experimentation with real systems are likely to be costly. It requires to interrupt the service of the system, and moreover if the changes made do not improve the process, this could lead into customer dissatisfaction. In simulation the cost are only the time needed to alter the model.

- Time: Experiment with real systems is time consuming since it may takes weeks or months before a true reflection of the performance of the system. Depending on the size of the model and the computer, a simulation can run many times faster so results can be obtained and analysed in minutes or hours.

- Control of the conditions: In order to properly analyse the results of the changes, it is useful to control the conditions under which experiments are performed. For example to avoid the Hawthorne effect where staff performance improves simple because some attention is being paid to them.

2.3.1 Types of Computer Simulation

The main objective of a simulation model is to give us the opportunity of observing a particular system. The system as it is describe in Air Force System Commands (1991) "*is a composite of people, products and processes that provide a capability to satisfy stated needs. A complete system include the facilities, equipment, materials, services, data, skilled personnel, and techniques required to achieve, provide and sustain system effectiveness*".

The idea of how to model a system depends entirely of the nature of the system that is the use of the model and how we perceive it. Here it is important to highlight that two people can look at the same thing and make completely different models depending on their personal world view. However when talking about systems we can classify them based on different criteria. First if stochastic or random behaviour matters, then we classify the system into stochastic or deterministic.

Regarding if the system changes with respect to time, we can define that a system is static if it does not change a lot in, if we need to consider how it evolves then it is called dynamic. Finally a dynamic system it is called also discrete if it changes at a discrete points of time otherwise it is called continuous [Ross10]. On the Figure 2-10 below the previously defined classification is presented, the green charts represent the classification of our model

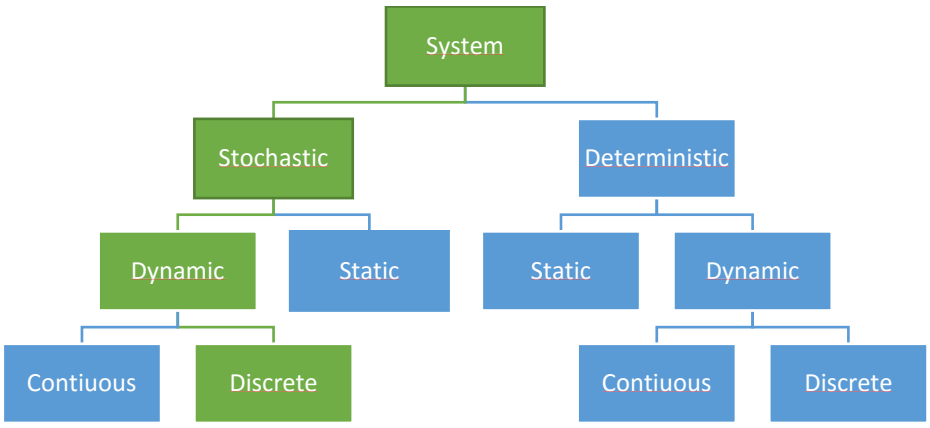


Figure 2-10 General types of systems

- Event Driven Simulation

A discrete event-driven simulation is a popular simulation technique. They are discrete and manage events in time. On other words, each event happen in a concrete point in time and changes the state of the system. Objects in the simulation model objects in the real world, and are programmed to react as much as possible as the real objects would react. A priority queue is used to store a representation of events that are waiting to happen. This queue is stored in order, based on the time the event should occur. As an event occurs, it can spawn other events. These subsequent events are placed into the queue as well. Execution continues until all events have been processed. Thereby, complex issues, such as the performance of production systems, logistic processes and the material flow can be simulated. In particular they enable to analyse the process chain without affecting the real production process [Geo14].

- Finite Elements Simulation

The Finite-Elements technology deals with predicting and optimising the behaviour of complex objects or systems that are normally interconnected. It divides the system analysed into a very large number of small (finite) volumes. FE solutions are commonly used to target potential solutions on different physical areas. In recent years, the FE method has been applied to structural mechanics.

The key advantage and benefit is the ability to see the properties distribution in a small scale providing a high level of understanding of the behaviour and connections between various factors affecting the system [Fin16]

Simulation Languages

Under discrete-event simulations, there are always a huge volume of computation, therefore, the need to use a computer to calculate this computations is essential. When implementing a simulation model some general-purpose programming languages are usually, FORTRAN, Visual Basic, C/C++ or Java which require above-average programming skills. However in the last days, and due to the increasing capacity of storage and computation of computers, some specialize languages have been developed for discrete or continuous simulations. It is common to find a trade-off between how flexible the language is in representing certain model situations, for

example more flexible programming languages require more work and care when developing the logic model. Some languages are programming oriented (e.g., SIMSCRIPT™) while others are more a “drag-and-drop” (e.g., ARENA™, Plant Simulation) [Ross10].

2.3.2 Simulation methodology

A methodology is series of steps to follow in order to achieve a goal, is this section is therefore presented the needed steps to solve a problem though systems analysis. The most general methodology is stated below as in [Ross10]:

1. Define the problem
2. Establish measures of performance for evaluation
3. Generate alternative solutions
4. Rank alternative solutions
5. Evaluate and iterate during process
6. Execute and evaluate solution.

However when we use simulation, some specific actions must be taken while performing the general overall problem solving and thus the general methodology should be refined. The whole methodology is presented in Figure 2-11.

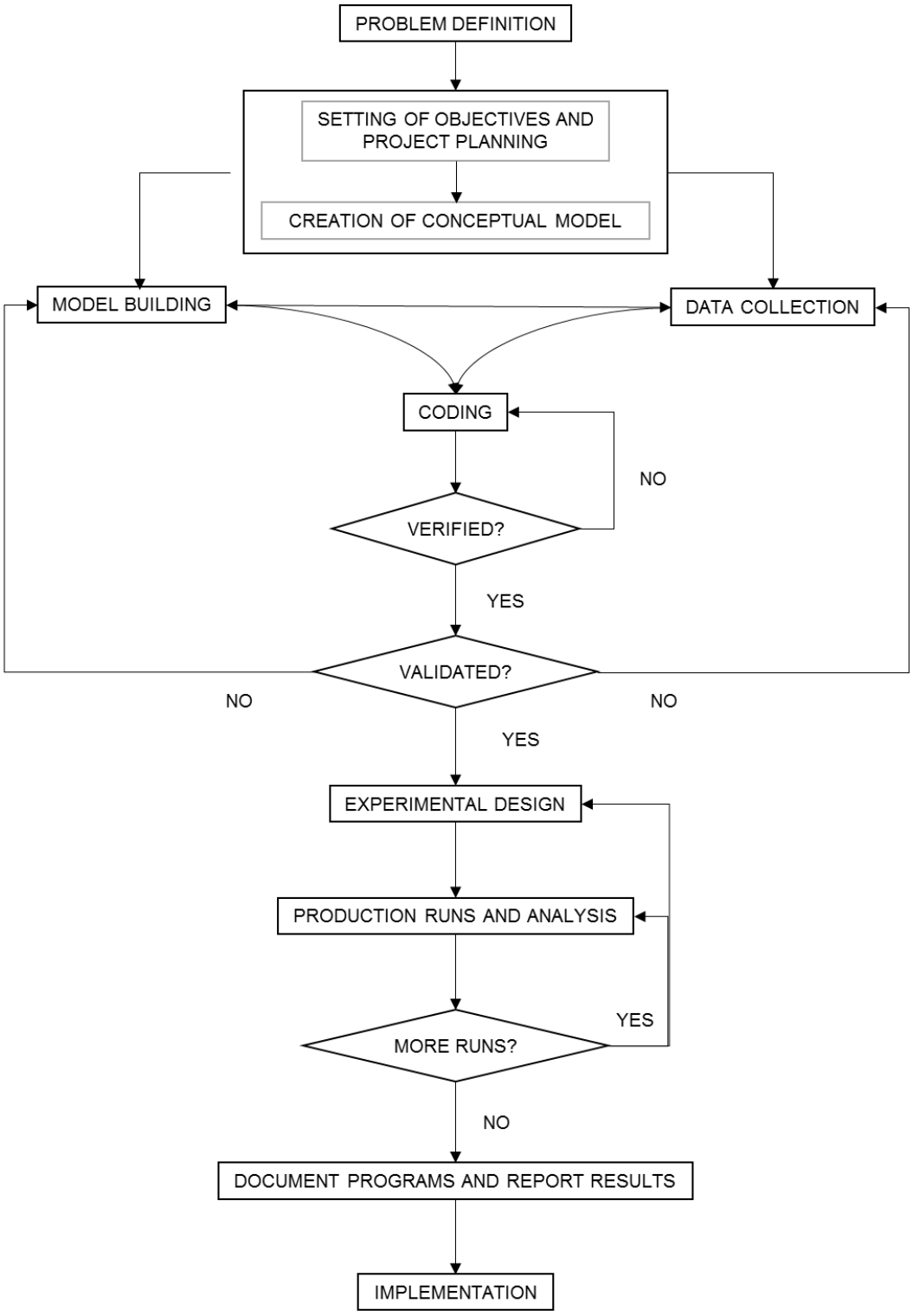


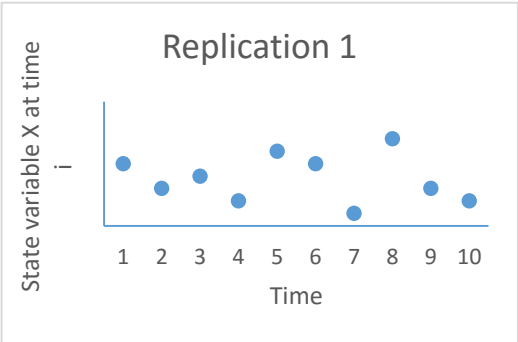
Figure 2-11 General Simulation Methodology [Ross10]

2.3.3 Analysing Simulation Output

In this section, we present a statistical analysis of the output of the simulation model in addition with some issues that arise from the proper execution of the simulation model like the initial conditions, or the input parameters. When referred to initial conditions, we want to resolve that if the system starts empty and idle, then we will need some time in order to reach the steady-state conditions under our simulations will be valid [Ross16].

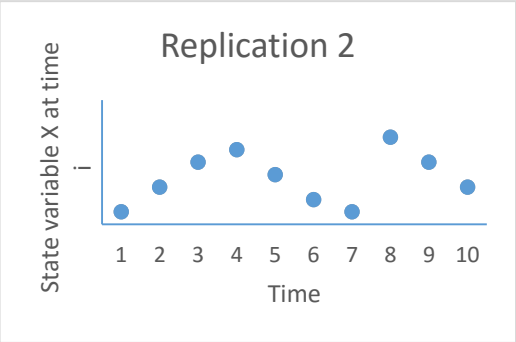
When an experiment on the simulation model takes place, the modeller sets the input parameter to the model and after that the model evolves over time until the termination point. At this point and depending on the kind of output that we want to measure some statistical quantities are shown in the output reports. An experiment may be for a single run of the model or may have more than one run.

Every single experiment define the unique initial conditions and its driven by the same input parameters, however, each replication is according to [Ross16] "*the generation of one sample path which represents the evolution of the system from initial conditions to its ending conditions*" thus, as the name implies, each replication is an independent "repeat" of the simulation. The concept it is depicted on Figure 2-12 and also two more ideas, first, the within replication statistics: "*statistical quantities collected during a replication*" and the across replication statistics "*Statistical quantities based on the observation of final values of within replication statistics*" on the example from Figure 2-12 the within replication statistic is the average of the variable X, and the across replication statistic is the average of the averages of the variable X and due to its nature, obviously only one value for replication is available [Ross16].



Within replication statistics:

$$\overline{X_1} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

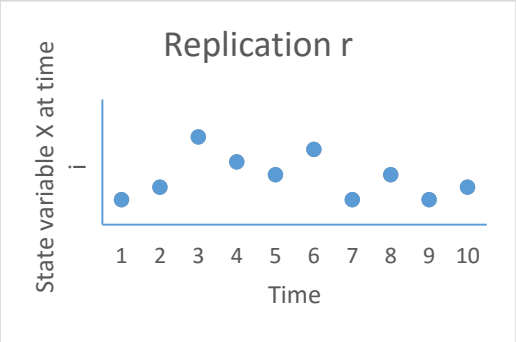


Within replication statistics

$$\overline{X_2} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Across replication statistics

$$\bar{Y} = \frac{1}{2}\sum_{j=1}^{2} \bar{X}_j$$



Within replication statistics

$$\overline{X_r} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Across replication statistics

$$\bar{Y} = \frac{1}{r}\sum_{j=1}^{r} \bar{X}_j$$

Figure 2-12 The concept of replicated simple paths.

Since the values are considered independent, then we can assume that the sample for across replication are also independent and identically distributed.

At the moment when a system is modelled, sometimes we require some specific measurement goals for the simulation responses, this goals, will give us an idea of how to carry out and analyse the experiments over the simulation model [Ross16]. In relation with the experimental analysis, we can find two categories regarding the period of time of the simulation, the definitions are both provided by Rossetti [Ross16].

- Finite Horizon:" *In a finite-horizon simulation, a well-defined ending time of ending condition can be specified, which clearly defines the end of the simulation. Finite-horizon simulations are often called terminating simulations, since there are clear terminating conditions"*

In finite-horizon simulations, the execution can finish whether there is a specific point in time or an event. It is important to mention that in case that some event

stops the simulation then it becomes a random variable. Each of the simulation runs represents a sample of the model. Some examples of this kind of simulation could be a bank that has its opening and closing time, a battle, where simulation last until the force strength reach a critical value or filling a customer order.

- Infinite Horizon: *"In an Infinite-Horizon simulation, there is no well-defined ending time or condition. The planning period is over the time of the system[…].Infinite-Horizon simulations are also called steady-state simulations because in an infinite-horizon simulation, you are often interested in the long-term or steady-state behaviour of the system*"

In contrast to the finite horizon, the infinite-horizon simulation has no natural ending point, however since our simulations cannot last forever we must define a specific simulation length that we hope is long enough to satisfy our goal of measure the long-run performance. Infinite-horizon are often linked to systems that operate continuously. One of the most important things in this systems when modelling them is to define what is meant by steady-state behaviour. Some examples of this type of simulation could be the emergency area of a hospital (open 24/7), telecommunication system or a factory where we want to measure the steady-state throughput [Ross16].

- **Point Estimates and Confidence Intervals**

In order to provide statistically robust results, in the model, the settings are tested by 60 simulations however and with the aim of reduce the number of simulations to the essentials an improvement over the present simulation model will be done. The idea is that the system automatically calculates the necessary number of simulations and stops when the desired level of confidence is reached. To set the background of this, some basics of statistical inference are presented.

When analysing the results from a simulation we have to know that we will find always an error in our estimation, the idea of the next section is therefore how to control this error level during the simulation experiment. The approach is try to figure out the number of samples necessaries to provide a certain level of confidence over the estimation

The main objective of statistical inference is to estimate, that mean that by studying a sample of the population we pretend to get valid information for the whole population

[Dav10]. A statistic is a function of the random variables in a sample, but usually there are some facts about population such as the size, the mean, the proportion of some attribute or the variability that investigators want to know, those “facts” are called parameters, therefore any parameter is a statistic used to estimate an unknown quantity based on the sample. However, this parameters cannot be 100% determined but can be estimated through using quantities called statistics.

We can find two ways of estimations for parameters:

- ➔ Point estimation: is a single value (value of a statistic) used to estimate a parameter, the value has been calculated from a sample [SHAY13].
- ➔ Interval estimation: it’s a range of values where we hope the parameter is contained.

There is another concept that we need to know because it will apply to our case and it is the central limit theorem [SHAY13] : “Regardless of the shape of the population, whether it was normal or not, the sampling distribution of $\bar{X}$, becomes approximately normal as n increases”

According to the definition provided in [SHAY13] *“An interval estimation is a range of values, calculated on the information of the sample that the parameter in the population will be within that range with some degree of confidence”*

To calculate this range of values we need to define first which the degree of confidence is. This level indicates the error in two ways [SHAY13]:

1. By the size of the range
2. By the probability of the true population parameter will be lying in that range

On the evaluation of the results of the simulation we will need to evaluate the precision as well as the reliability of the results, the idea here is to control the sampling error among the experiment. The approach we will use is to determine the number of samples so we can know that our results are robust enough [RosD16].

T

So we all know that the average of the sample used to be a good way of estimating the average of the population thus a good estimator for $E[X] = \mu$. The sample average has already been shown as an example for within replication statistics in Figure 2-12 and it is presented again in equation (2)

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{2}$$

$\bar{X}$ is a function of many random variable from the sample so it is as well a random variable, this condition provides $\bar{X}$ the property of being an statistic. There is a probability distribution for each random variable and we can define a sampling distribution as the probability distribution associated with each statistic. This distribution can be used to create a confidence interval on the point estimated linked with the statistic. The point estimate for the sample average calculated from the sample is (3)

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3}$$

The degree of certainty linked to a point estimate is expressed by a confidence interval, this interval does not imply that the parameter μ is inside the interval, actually it could lay inside or not. We should rather think into the confidence level 1-α when computing the interval. The value α can be defined like the risk that the confidence interval procedure create a specific interval that don't contain the true parameter. We will use the confident interval to evaluate the risk of making a bad decision.

- **Testing of Statistical Hypotheses**

The testing of hypothesis is a process that leads to the acceptance or rejection of a null hypothesis. The process first set up a Null Hypothesis and an alternative and therefore we will have the option of reject the null hypothesis or not reject the Null hypothesis (it does not mean that its true)

During decision making, and given that the true is never known, we can make two types of errors:

- ➔ Type I Error: It takes place when the test reject the null hypothesis when it is true, the probability is given by α.
- ➔ Type II Error: It takes place when we do not reject the null hypothesis when it is false. It is denoted usually by β [SHAY13].

It is found on the literature that α is commonly called level of significance, the most common values are 0.01, 0.05 or 0.1.

As mention before, we assume that the sample size is large enough to accomplish the central limit theorem, so the distribution of $\bar{X}$ is close enough to a normal [RosD16]. Then we can compute a confidence interval based on the point estimate, $\bar{x}$. the sample variance is (4):

$$s^2(n) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{4}$$

And it is also a good estimator for the Var[X] = σ^2, finally we can calculate the confidence interval for μ like you can see in (5):

$$\bar{x} \pm t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \tag{5}$$

Where $t_{\alpha/2,n-1}$ is the upper 100(1-α/2) percentage point of the Student t-distribution with n-1 degrees of freedom and n is the number of samples.

The document presents now some different possible methods to calculate the number of replications needed to acknowledge a certain level of confidence and provide robust enough results .The motivation to do this is to provide an easy-to-use method that will be included into existing simulation software to allow practitioners to get result within a specific accuracy. One of the problems faced here is that current simulation software provides little or even no guidance to user at the time of making decision such as warm-up period, run-length and number of replications [Hoa07].

The question of the number of simulation runs that need to be done has not been systematically addressed. The intuitive sense tells us that the higher the number of simulation runs the better, 100 simulations runs will give us more trustable results than 5 however is it possible that 5 simulation runs are enough? Or is it possible that 100 is not enough? Which is the number of simulation that can allow us to say “my model’s

predictions are an accurate representation of the real model"? Running the model from an infinite population is sampling from an infinite population, this lead us to the necessity of accept some uncertainty about the prediction of the model [Byr13]. Some different methods to address this problem are presented now in order to provide a minimum number of runs based on power calculations when the runs are expensive or just to provide a maximum in case that simulation runs are inexpensive.

The way of facing the problem of calculating the number of simulation runs depends on the horizon of the simulation. If we are facing a Finite Horizon then we can find the following methods [Ross16]:

i) Iterative method based on t-distribution
ii) Approximate method based on the normal distribution
iii) Half-width ratio method

- **Iterative method based on t-distribution**

A student's t-test is any test where the used statistic have a t of student distribution if the null hypothesis is true. It is used to determine when the population follows a normal distribution but the size of the sample is not big enough to make the statistic where it is based on normally distributed. We used an estimation of the standard deviation instead of the real value. We will use it to see if the average of a normally distributed population have a value specified on the null hypothesis.

Since the model is complex, and the variance of the population is not known, we need to ask if the population is normally distributed, if it is not we need to do a non-parametric approach but if it is, then we can build the confidence interval around μ.

Applied to our case, at the end of the first n number of runs of the simulations we will have for example n number of mean processing time for the type of component I. The analysis of the system requires a number of replications to obtain an output result for decision making with a degree of confidence of 95% or 99%. We will use the confident interval for a point estimator as a base for determining the size of the sample, from the last equation (5) we can say that:

$$h = t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \quad (6)$$

This h is called half-width of the interval, later we should pick a limit. This limit or bound [E] is the error that we want to assume when computing the confidence interval and finally pick a sample that satisfies (7):

$$h = t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \leq E \tag{7}$$

Where $t_{\alpha/2,n-1}$ is the upper (α/2) percentage of the t-distribution with n-1 degrees of freedom. We want to know the value of n, however, the t distribution also depends on n so this previous equation is iterative and gives the name to the method. Thus, if the condition is not reached then we need now to increase the value of n by one unit, recalculate everything and check again if the condition is fulfilled. This process would last until the condition is reached. This method is the one chosen by the IFA institute and it is already implemented on the case of train couplings. Since its results have been already checked and proved this method will be also implemented on the case of smart wheel set.

- **Approximate method based on the normal distribution**

This method is also based on previous calculations, in this case the required sample size can also be calculated by an approximation to normal distribution, by finding n from the last equation (7) we can compute (8):

$$n \geq \left(\frac{t_{\alpha/2,n-1} * s}{E}\right)^2 \tag{8}$$

As n gets bigger and bigger the value of $t_{\alpha/2,n-1}$ converge to the upper percentage 100(1-α/2) point of the standard normal distribution $z_{\alpha/2}$. This leads us to the following approximation (9):

$$n \geq \left(\frac{z_{\alpha/2}}{E}\right)^2 \tag{9}$$

According to Rossetti [RosD16], this equation usually works for n>50, however in our case and we since that have check that we can achieve a certain level of confidence after about 30 simulation we will intend to achieve to run less than 50 times so the method could be implemented and evaluated but the potential is not high.

Both previously presented methods require an initial value for computing the standard deviation, after calculated only the degree of confidence and the bound remains but they are up to the modeller. It should take into account that the lower the E the higher the amount of replications needed with a quadratic expression but as said before it is something under subjective control.

- **Half-width ratio method**

When carrying out simulations with other software such as Arena, it is quite common to use the half-width method since Arena does not repot the standard deviation but the directly the half-width value for a 95% of confidence. If we were using Arena we would only have to use the directly reported h_0 after a previous simulation run of n_0 replications.

Since the software we use it is Tecnomatix: Plant Simulation, we will have to compute the half-width as follows (10):

$$h_0 = t_{\alpha/2,n_0-1}\frac{s_0}{\sqrt{n_0}} \tag{10}$$

Then solving n_0 in (10) yields (11):

$$n_0 = t^2{}_{\alpha/2,n_0-1}\frac{s_0^2}{h_0^2} \tag{11}$$

Following, for any n, the equation is (12):

$$n = t^2{}_{\alpha/2,,n-1}\frac{s_0^2}{h_0^2} \tag{12}$$

Then making the ratio in between n and n_0 ((11) and (12)) and assuming that $t^2{}_{\alpha/2,n_0-1}$ is approximately equal to $t^2{}_{\alpha/2,,n-1}$ and that s^2 is also equal to s_0^2 the ratio yields (13):

$$n \cong n_0\left(\frac{h_0}{h}\right)^2 \tag{13}$$

All the previously presented methods are tested on an example in [Ross16], the results present that the easiest method to compute is the half-width method since the value is already given by Arena. However and due to the nature of the assumptions made in the method, this third method leads to a higher number of replications that the others. Since our software it is not Arena, we will have to compute the value of the half-width method

It is shown in [Byr13] that the use of inferential statistic it is not recommended in the attempt to show “no significant difference” between the real and the model data. The reasons for the no recommendations are first, that the model failures to reject with an inferential test does not bring us enough evidence that two samples are equal. Second that the test fail when we try to find a difference and small sample sizes are used. The model gives us high uncertainty levels under low number of samples or simulation runs.

- **Methods of Proportions and Interval Data**

The idea of those methods is to use the opposite strategy as the computation of confidence intervals. In confidence intervals we usually calculate the sample mean and then construct the interval around it, given the desired confidence level [Byr13]. The strategy to follow is to computes the appropriated number of simulation runs in order to get the desired confidence interval with a desired confidence level. As well as in the previous methods, two critical assumptions must me made:

- Random sampling: The runs do not introduce any systematic bias (some random numbers are more likely than others). It depends on how random numbers are generated and it used to be unlikely unless that poor generation strategies are followed.

- Independent and identically distributed: Each run it is independent from the others. That is, what happen in one of the simulation runs does not affect the others and all the runs should come from the same kind of distribution. This last condition can be particularly problematic in our case when dealing with pooling, on each run the same strategy should be followed since they will affect the output.

- **Method of proportions**

It is presented in [Byr13] a method to calculate the number of simulation runs based on proportions. The most common form of this proportions are if the trial is correct or not. The standard equation for a confidence interval based on proportions is:

$$CI = p \pm z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} \tag{14}$$

Where CI is the confidence interval, p is the observed proportion, n the sample size that we want to calculate and $z_{\alpha/2}$ is the value of the normal distribution that have the upper tail set by a given confidence level. The size of the confidence interval (w) based in the proportion p is given by:

$$w = z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} \tag{15}$$

From this equation the required number of simulation runs can be easily solved as follows:

$$n = p(1-p)\left(\frac{z_{\alpha/2}}{w}\right)^2 \tag{16}$$

Any integer number above n should satisfy our confidence interval. One of the problems of this method is that it depends clearly on the proportions p. On [Byr13] an example to show this problem is presented, when p is close to 0 or to 1, the number of simulation runs gets smaller. To solve this, the Yates correction it is applied. This correction says that if np<10 then another formula must be used.

$$w = z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} + \frac{0.5}{n} \tag{17}$$

Solving it again for n:

$$n = \frac{\frac{z\propto/_2}{w} + p(p-1) + \sqrt{\left(\frac{z\propto/_2}{w} + p(1-p)\right)^2 - \left(\frac{z\propto/_2}{w}\right)^2}}{2\frac{z\propto/_2}{w}} \quad (18)$$

The method presented above, need proportions as their output, however, since our model does not fulfil this condition it will not be implemented.

- **Method of Ratio or Interval Data**

The method of proportions is easier and simpler to work with since we cannot find issues with the units. In the case of this method and in some of those presented before the meaning of an interval width is dependant of the units. ±1% has a clear interpretation in proportions but ±100 units, for instance, depend on the unit and maybe of the scale of the data. For example, saying that ±100 ms when the average is 300ms, does not seems accurate, but if the mean is 10 minutes then we may say that we have enough precision.

To apply this method, some simplifications are made. The first one is that the width (w) is a proportion of the mean, this allow CI to be compared across measurements and simplifies equations. So if the mean tardiness of a sample is 2days and de desired error is ±0.1 days then w=0.1/2=0.05. The second simplification it is made with the variance. The equation for a confidence interval for ratio data is:

$$CI = M \pm z\propto/_2 \frac{s}{\sqrt{n}} \quad (19)$$

Where again CI is the confidence interval, M is the sample mean, z is the value of the standard normal, *s* is the standard deviation and n our desired sample sized. The size of the interval is a matter of the standard deviation (square root of variance), the degree of confidence we want and the number of simulation runs. In some models like in modelling human performance, the standard deviation scales with the mean, this means that longer tasks will have also longer standard deviations. Based on this, the *coefficient of variation (CV)* is calculated in order to measure the variability as follows:

$$CV = \frac{s}{\mu} \tag{20}$$

If the standard deviation and the mean scale at the same ratio, then CV remains constant. Since w is a multiplier based on the mean, then the width w can be calculated by solving this equation:

$$w\mu = z_{\alpha/2}\frac{s}{\sqrt{n}} \tag{21}$$

Solving for n and substituting σ/μ for the coefficient of variation:

$$n = \left(\frac{z_{\alpha/2}}{w} CV\right)^2 \tag{22}$$

In the practice, this method is assuming that we know the standard deviation of the population. This is fine as long as we have a sample size quite big (over 100).If we have smaller values of n then we have to use the t-distribution rather than the normal (z) distribution. The problem now is: how can one know the CV of a model prior to running it? the answer presented in [Byr13] seems to run 15-20 times in order to calculate the CV and then calculate n again.

3 TECNOMATIX PLANT SIMULATION

3.1 Introduction

Once that the importance of using simulations have been presented, on the state of the art and in order to introduce the software used to model and simulate the MRO process, the thesis presents a brief description of the capabilities of the software. Since the thesis will be based more on improving the present simulation model rather than modelling a new one, the description will present the shortly the functions for each element of the software. The first thing we can see when we open the software (Figure 3-1) is the console, the frame window, the toolbox and the class library, all of this features will be described in the next part.

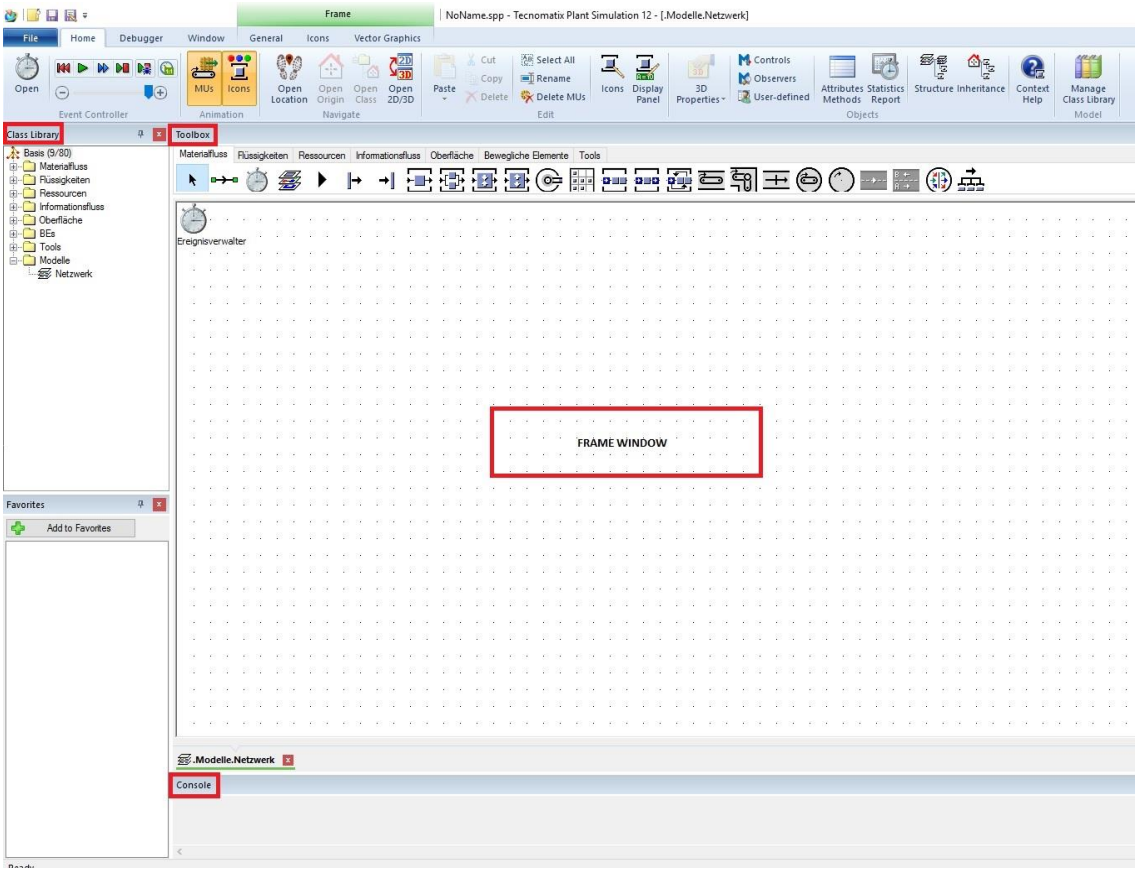


Figure 3-1 Overview of plan simulation

This first window will be our main workplace where the model can be designed by using a drag-and-drop style with the different elements

- **Frame Window:** Is the area where the simulation is designed. Here the different objects from the simulation are dragged and connected among them.

- **Console**: As it is defined in the [Bang10], the console "*provides information during simulation (e.g. error messages)*". We have to option to print output messages of the simulation in the console. If the console it is not needed we can hide it.
- **Toolbox**: This area is aimed to provide "quick access" to the class library, to the categories because it is customizable. It shows the tabs for material flow, fluids, resources, information flow, user interface, mobile units and tools.
- **Class Library:** Here the user can find all the objects necessary for the simulation. It is possible to create own folders, duplicate classes and give them their own properties, create different frames and hierarchy levels or load objects from other simulation models. The class library can be edited by adding or subtracting objects from it.

Now in order to properly present the software and the potential that it has, every component will be briefly explained starting with the different classes that we can find in the library, then the main icons of the toolbox and finally a short example of model where we will show how the software work and highlight some of the possibilities that it allows.

3.2 Standard Classes

A class, according to [Bang10], is a „used-defined data type", created to define a concept that has no direct equivalent in the basics data types. In other words is an object that it is different from the standard objects. Each manifestation of the class it is called instance of the class and it inherit the basic properties of the class with some special characteristics (like another name). In plant simulation the classes are divided into seven different categories, as shown in Figure 3-2:

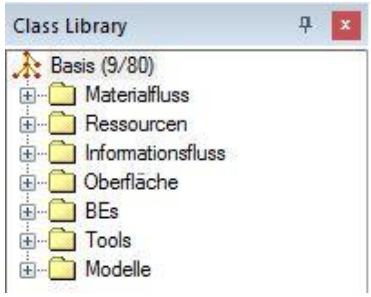


Figure 3-2 Class library

3.2.1 Material flow objects („Materialfluss“)

In the first category we can find mobile and static flow objects. The mobile units (BEs) or MUs from now on, are explained later but by now we can say that they represent the physical or logical objects that are moved all along in the model. These mobile unit are transported by active or passive objects. The difference among them is that the active objects such as Single Station (“Einzelstation”), Parallel Station (“Parallelstation”) or Assembly Station (“Montagestation”) dynamically move the mobile units though the connectors. The source and the drain (Quelle und Senke) are the limits of the model and they are in charge of create and destroy MUs. Finally the passive objects such as Store (“Lager”) or Track (“Förderstrecke”) do no move the Mus automatically and thus need a method if we want to remove the MU from the object.

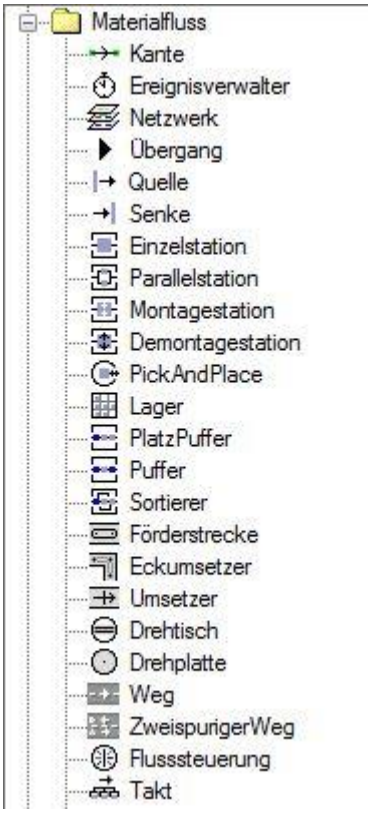


Figure 3-3 Material flow objects

The different elements from the material flow folder and their description are presented on the table below. Notice that since the software and therefore implementation of the method it is in German, the names of each icon will also be in German, however a

translation of the components into English it is provided in order to make the reader's comprehension easier.

Icons	Name of the object	Description
	Connector ("Kante")	Allows transfers between objects (it can be more than two)
	Event Controller ("Ergebnisverwalter")	Controls the simulation beginning, end and speed
	Network("Netzwerk")	It is an object that allows us to create different hierarchical levels (sub-processes)
	Transition("Übergang")	Allows connections between hierarchical levels
	Source("Quelle")	Creates MUs under designer's specifications (what MU or when to produce it)
	Drain("Senke")	Destroys MU after processing them, also collect statistics as throughput time or number of parts destroyed
	Single Process ("Einzelstation")	Accepts only one MU. Simulates a single job or machine
	Parallel Process ("Parallelstation")	The operation is the same as Single Process but works with more than one MU and allows different processing times
	Assembly station ("Montagestation")	Allows to simulate assemblies between parts (eliminate input MUs and creates a different MU)
	Dismantle Station ("Demontagestation")	Allows to simulate dismantlement (eliminate input MUs and creates two or more different MUs)
	Store("Lager")	It simulates a warehouse, allows to decide the capacity(organized like a matrix)

	Buffer (“Puffer”)	Collects MU and allow to choose the exit strategy (FIFO,LIFO…) and the capacity
	Sorter (“Sortieren”)	It allows to reorder the located MUs with different criteria. We can define capacity, order or sorting criteria
	Flow Control (“Flussteuerung”)	Model strategies for splitting-up and bringing together the flow materials
	Cycle (“Takt”)	Synchronizes the transfer of parts from station to station

Table 3-1 Material flow objects

3.2.2 Resources („Ressourcen“)

The resources folder is intended to provide the necessary objects to simulate human capital or workers. This use can be especially useful in repairs (a worker would go the make the maintenance), machine operations (some machines need an operator to use them, or maybe just to supervise) or employer that transport parts. In this folder all the elements to model this processes are provided. In Figure 3-4 we can see the different icons and later on in Table 3-2 the description of each one of them.

Figure 3-4 Resources in Plant Simulation

Icons	Name of the object	Description
	ShiftCalendar(“Schichtkalender”)	It allows to fix shifts, holidays, working days…

Table 3-2 Resources’ Description

3.2.3 Information Flow („Informationfluss“)

This objects have the aim of manage, create or export information and data in the model. We could classify them into 4 categories, list and tables, Trigger and generator, AttributeExplorer, Objects for data exchange. In the Figure 3-5, the different objects found in the Information flow folder are depicted and in the Table 3-3 Description of the information flow objects are the descriptions of the most used.

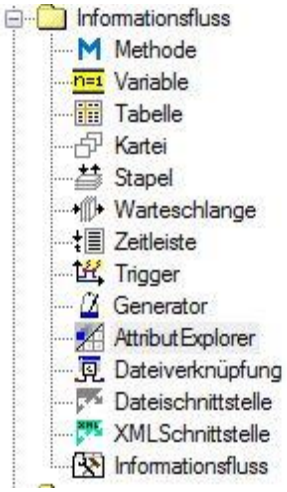


Figure 3-5 Information Flow objects

Icons	Name of the object	Description
	Method (“Methode”)	The methods are the objects used to set and control the model, in general are able to get information from other objects and return a value, calculates a value, set parameters or actions or in general control behaviour of other objects.
	Variable (“Variable”)	Allows to see a variable in the frame window, it can be controlled from a method.

	Table (“Tabelle”)	Two-dimensional list, it have many applications, since information storage, production orders, statistical information…
	CardFile (“Kartei”)	One-dimensional list with random access through an index. It can be used as an array, useful to storage same type of data
	Stack File (“Stapel”)	One-dimensional list accessed according to FIFO or LIFO principles
	Queue file (“Warteschlange”)	One-dimensional list accessed according to FIFO or LIFO principles
	Timeline (“Zeitliste”)	Allows to record and manage temporal values progressions like stock at a fixed point in time
	Attribute Explorer (“AttributExplorer”)	It allows to manage all the attributes from different object in one location

Table 3-3 Description of the information flow objects

3.2.4 Display and User interface objects („Oberfläche“)

The objects presented in the folder User Display allow the user to read and also analyse the output data of the model. It allows the user to create charts that show dynamically the results (plots, bar diagrams…) during the simulation or dialog windows in order to easily change parameters or attributes from the different objects. In the Figure 3-6 , the different objects found in the Information flow folder are depicted and in the table are the descriptions of the most used.

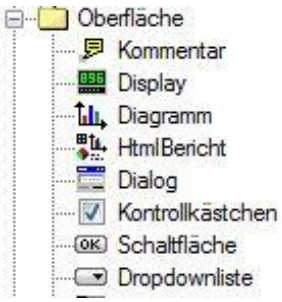


Figure 3-6 Display and User interface objects

Icons	Name of the object	Description
	Comment (“Kommentar”)	It is used to describe the simulation model or add some clarifying
	Display(“Display”)	It is used to show a value or just data during the simulation. Needs a method in order to reset the value
	Chart (“Diagramm”)	Displays the data in several forms. Data can be plotted from a table, from an input channel or just an attribute from the object. It has a Statistics Wizard to easily plot the results.
	Report (“HtmlBericht”)	It creates a HTML page that can be opened in any HTML browser
	Dialog(“Dialog”)	Allows to provide simple user interface for complex simulation models or prevent the user from modifying the Frame. It creates only one dialog window
	Checkbox (“Kontrollkästchen”)	Insert a checkbox into the Frame that can be used to toggle between states *on/off,* operating modes…
	Drop-down List (“Dropdownliste”)	Inserts a list that can be defined by the user, when something is selected, the control is executed
	Button (“Schaltfläche”)	Inserts a button into the Frame that executes an action programmed in the control

Table 3-4 Description of the user interface and display objects

3.2.5 MUs („BEs“)

The MUs (Mobile Units), or BEs (Bewegliche Elemente) in the German version, are the objects used to model the material flows in the simulation model. Depending on the object different features can be adjusted, for instance, we can define the length, width, height, speed or acceleration of a transporter. For each MU we should also define the booking point that is the position of the MU. Here the different MUs and their description are presented:

Figure 3-7 MUs

Icons	Name of the object	Description
	Entity (“Fördergut”)	It models the different parts that are transported, processed , assembled

Table 3-5 Description of the MUs

3.2.6 Tools („Tools“)

Some other tools are available on the software in other to help to analyse and optimise the processes, with them we can also get statistic results. For instance with the BottleneckAnalyzer it allows us to find bottlenecks and decide where a buffer could be necessary. In the default version only the BottleneckAnalyzer, Sankey Diagram, Energie Analyzer and TransferStation are shown but more tools can be easily added from the Class Library Manager, some of them are shown in Figure 3-8 and described in Table 3-6.

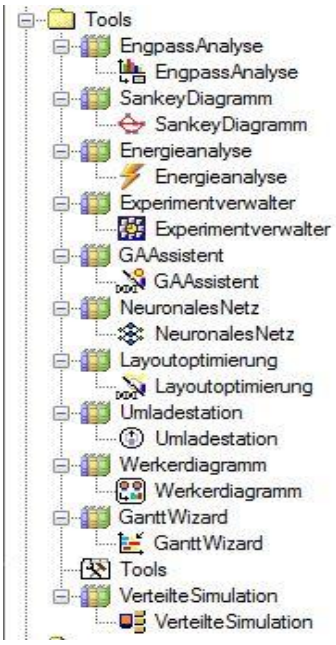


Figure 3-8 Other Tools from Plant Simulation

Icons	Name of the object	Description
	BottleneckAnalyzer („EngpassAnalyse“)	Provide standard statistics of the material flow objects in a table. Works statically (do not provide results during run)
	ExperimentManager („Experimentverwalter“)	Allows to set the number of simulation runs as well as to carry out experiments by selecting one or more input variables or attributes and an output that will be evaluated.
	TransferStation („Umladestation“)	Useful object able to load, unload, reload and move parts from one object to other without connectors. It is used to transfer parts among the different hierarchical levels

Table 3-6 Description of the Tools

3.3 Frame and Commands Window

The frame window give us a bunch of options used to make the design of the model easier, in previous versions of the software there were some icons related to the control of the frame window however in the latest version of the software (Tecnomatix Plant Simulation 12) this icons have been merged with the command window. Therefore only some icons that are related with the frame control or the simulation will be described in Table 3-7.

Icons	Name of the object	Description
	Simulation-Speed Controllers	Allows the user to control the speed, play, stop, finish the simulation or even simulate at real-time
	Zoom In	Increase the size of the frame
	Zoom Out	Decreases the size of the frame
	Original size	Gives the frame to its original size
	MUs	Switch on/off the animation of mobile units
	Open Location	Opens the location of the frame where the frame is locate, if it is a class then opens the class
	Open Origin	Open the origin from which the selected object is derived
	Open Class	Opens the class of the selected object
	Structure	Show the object which the selected object contains, if no object is selected it shows the structure of the opened frame
	Inheritance	Shows the objects which inherit from the selected object

	Manage Class Library	Opens the manager that allows the user to select which libraries want to have available. Some libraries are only available under certain licenses.
	Find Objects	Opens a search manager that allows to search any object in the present frame or in sub frames
	View Options	Allows to show or hide MU names, object labels display panels, successors, predecessors or animation points

Table 3-7 Description of icons from the command and frame window

The debugger's tab (see Figure 3-9) contains a group of tools that allows the user to find mistakes while programming methods, it allows to debug the simulation model and do things like stop the simulation and check how the simulation is proceeding. We can also set breakpoint to stop the simulation and therefore monitor the methods that the used have programmed.

Figure 3-9 Debugger Tools

The debugger provides access to all data involved in methods Due to the ultimate objective of this project, which is the implementation of the validation methods in the software it is considered extremely important the tab present in the first window of the software called debugger, thus, it is also deeply explained in the Table 3-8 below.

Icons	Name of the object	Description
	Switch to Debugger	Open the window of the Method debugger, where we can go line by line checking the code

	Stop on Controls	This command stops the simulation when the Event Controller finds a Methods started by an object before executing the calling method and shows it in the Debugger.
	Stop on Errors Handlers	The simulation is stopped when an error handling method is evaluated
	Stop on Wakeup	The simulation stops and open the debugger when the execution of a suspended methods is continued
	Stop on Formulas	With this command the simulation stops when finds a formula and open the Debugger.
	Ignore Breakpoints	The software ignore any user-defined breakpoints
	Ignore Errors	This command allows the user to continue the simulation although the model contains faulty code.
	Ignore Formula Errors	The command prevents Plant Simulation from opening the Debugger when a runtime error occurs in a formula.
	Delete Breakpoints	It deletes all breakpoint in all methods

Table 3-8 Description of the debugger tools

3.4 SimTalk

The basic tools of Plant Simulation are not enough to generate a real model. In order to provide the software this flexibility and extend the standard features Tecnomatix Plant Simulation have the SimTalk programming language. It can be divided according to [Bang10] into two parts:

1. Control structures and language constructions such as loops or conditions
2. Standard methods connected to material flow objects. They are already built-in and have the basic functionality.

Naming the method

The programming language SimTalk does not differentiate between upper- and lowercasing in names or commands. In the case of the names some specials are already defined:

- Reset: It is executed when pressing the reset button
- Init: It is executed when the simulation starts
- EndSim: It is simulated at the end of the simulation

General Structure of a Method

We the user open a general method there is always a structure that need to be followed. First of all, the declaration of the different variables is done between the is-do and later on the code, with loops or conditions is introduced between do-end. After each line of code a ";" must be placed.

```
is
     Here the user should write the variables
do
     Here comes the code
end;
```

3.4.1 Variables and Data Types

A variable according to [Bang10], is „*a named location in memory*“. The idea is that this variables have a meaningful name. In Plant Simulation we can distinguish between local and global variables.

- Local variables: Are those variables that can only be accessed from the respective method where they are called. It cannot be read or modified from other methods. They are declared in between the “is-do” section of the method. The declaration is made by writing the name of the desired variable, colon and the data type as shown in the example below:

```
is
Name : Type;
do

-- statements

end;
```

Depending on the type of variable declared, the systems saves more or less space in the memory. The variables can be converted after being declared and their initial value is set by default.

- Global variables: This kind of variable are visible from any part of the model. All methods (in every frame) can access to them and modify them. To declare a global variable it must be done by adding the object “Variable” from the material flow folder to the frame. To read or modify the variable it is enough to write their name if we are in the same frame or the whole path if it is in another frame. Global variables can set its initial value (value when reset is pressed).

SimTalk provides the data types that are listed in the following
Table 3-9 [Bang10].

Name	Range of Values
Acceleration	Real, m/s^2
Any	The type is fixed after the assignment of the value
Boolean	TRUE or FALSE
Integer	-2.147.483.648 until 2.147.483.647
Real	Floating numbers
String	Character (letter or numbers)
Object	Reference to an object
Table	Local variable with table behaviour
List	Explained in Table 3-3
Stack	Explained in Table 3-3
Queue	Explained in Table 3-3
Money	Currency (€, $...)
Length	As real, interpreted in meter but can be changed in the settings
Weigh	Real, Kg normally
Speed	Real, measured in m/s
Time	Real, the smallest unit is the second and the structure is <dd>:<hh>:<mm>:<ss.sss>
Date	Dates from 1.1.1970 to 31.12.2038

Table 3-9 Data Types and Range Values

3.4.2 Operators

Through the combination of different variables or attributes we will be able to define complex expression that represent the real systems. In Plant Simulation we can find three types of operators [Bang10]:

1. Mathematical operators, used to operate with number variables. The different available mathematical operators are shown in Table 3-10.

Symbol	Use
+	Addition, link between strings
-	Subtraction

*	Multiplication
/	Division
//	Integer Division
\\	Module (remainder of integer division)

Table 3-10 Mathematical Operators

2. Logical operators, allows the user to establish comparison that lead to a Boolean result. The different logical operators are shown in Table [Bang10]

Operator	Function	Result
AND	Logical AND	TRUE if all expressions are true
OR	Logical OR	TRUE if at least one expression is TRUE
NOT	Not	Invert the Boolean value
<	Less than	
<=	Less or equal than	
>	Greater than	
>=	Greater than or equal	
=	Equal	
/=	Unequal	

Table 3-11 Logical Operators

3. Assignments operators, the last type of operator, allows the user to assign a value to the variable. It is done by the operator ":="

3.4.3 Decision-making structures in SimTalk

When the complexity of the modelled system increases, we need to execute instructions based on certain conditions. To achieve this objective, the software provides the basics of programming

Branching

The branching allows the method to carry out an action depending on a logic condition. It decides if the next instruction is executed or not. If the condition is fulfilled (TRUE), the "if" branch is executed, otherwise the else branch will be performed. In Plant Simulation we can find two types of branching:

- One-Condition branching: Only one decision is evaluated and some instructions are executed depending on the result of the condition .An example of this type of branching is presented below.
- Bound Branching: More than one decision is evaluated, the “if” the condition is met, the next condition is checked. It is useful when programming conditions with interdependences.

```
is
     local
          value1:integer;
do
     value1:=12;
     if value1<10 then
          print “If branch executed”;
     else
          print “Else branch executed”;
     end;
end;
```

Case Differentiation

This structure allows the user to program instructions that depend on one condition, nevertheless, the difference with Bound branching lie in that Case differentiation does not incur in hierarchical levels of differentiation. Hereunder an example of this type of structure is depicted.

```
is
     local
          num:integer;
do
     num:=2;
     inspect num
```

```
            when 1 then print "Num is 1";
            when 2 then print "Num is 2";
            when 3 then print "Num is 3";
      else
            print "The number is different";
      end;
end;
```

Loops

The loops structure are useful when the user need for example to evaluate values of a list of table. Depending on when the decision is made, the loops can be classified into two types:

- Header-Controlled Loops

When the system reach the loop check a condition first, if it is met, then the loop is executed. The conditions of the loops are executed while the conditions is repeatedly met. The idea is that the instructions of the loops change the value of the conditions, otherwise the systems enter in an infinite loop. Hereunder and example of the structure.

```
is
      num:integer;
do
      num:=2;
      while num<10 loop
            print "Loop number:" to_str(num);
            num:=num+1;
      end;
end;
```

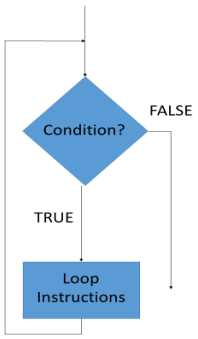


Figure 3-10 Header-Controlled Structure

- Footer-Controlled Loops

In the footer controlled loop the condition is checked after the first execution of the loop. This structure ensures that the loop is done at least once. One major difference against the header –controlled loops can be seen in Figure 3-11 Footer-Controlled Structure, the systems exit the loop when the condition is met.

```
is
     i:integer;
do
     i:=2;
     repeat
          print "Loop number:" to_str(i);
          num:=num+1;
     until i>3;
end;
```

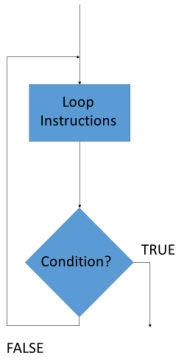


Figure 3-11 Footer-Controlled Structure

One more type of loop can be used in Plant Simulation, this is the For-Loop. It is useful when the user knows how many runs the loop has to be executed. Usually a running local variable is declared in order to control the loop runs.

```
do
     for <initialization> to <end value> loop
          -- Loop instructions
     next;
end;
```

4 DEVELOPMENT OF METHODS

4.1 Introduction

This chapter of the written work is based on the previously presented software knowledge; on it, the methodology to implement the different methods to validate the results of the simulation output are explained. The chapter begins with a brief presentation of the structure followed to implement the method, then a description of the variables that will be measured, and the variables that the user need to provide for each validation method and finally the development of the method for each kind of validation strategy is explained in detail.

The first thing that we should take into account when implemented the validation methods and since our primary objective is to get a replications number based on statistical results, we should know that the object in charge of providing the number of simulation runs is the Experiment Manager or “Experimentverwalter”. This object allows us to carry out experiments and then provide statistical reports across the replications as explained in Table 3-6. The object provides support in executing simulation studies and even distribute the work load among other computers if the simulations are large. However in the simulation model, the number of simulation runs is already present as “Durchlauf”. This variable appear in two methods of the model when computing some variables after each simulation run. This methods will be slightly modified in order to achieve our results.

The general structure of the information flow is depicted on Figure 4-1 , here we can see that the first step is to collect the variables that we are called “user-provided”. The system make an initial experiment and we will collect the result of each replication on a table. After this, once that we have initial data from to work with, the method calculate the number of replication runs necessaries to achieve a certain degree of confidence and with the fixed error and also will plot the confidence interval around the mean depending on the replication number.

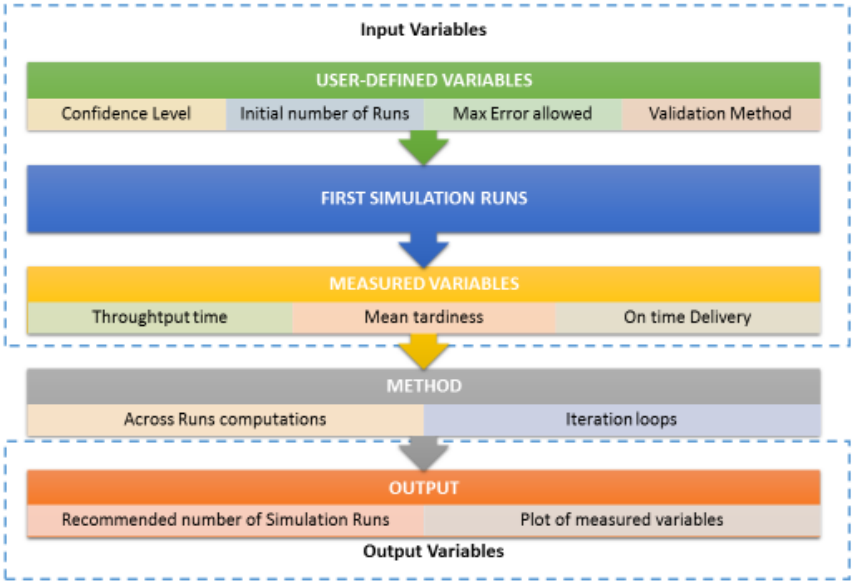


Figure 4-1 Information Flow Structure

4.2 Input variables

The input variables are divided in two types, first of all user-defined variables provided by the person in charge of validating the output of the simulation model and later on, once that several runs have been done, the mean or total values of some variables are recorded. It is important to consider that both types of variables are used when computing the number of simulation runs.

4.2.1 User-defined Variables

Since all the investigated validation methods are based on statistical inference, some parameters need to be user-defined. In order to make the use of the system as easy as possible for the user and also avoid them from possible modifications of the code (that could lead to undesired bugs), the software will show a dialog window where the software allows the user to easily define the parameters for the validation and the decided the method itself. All the user-defined variables will be defined as global variables in order to be able to control the in the different levels of the hierarchy. The input variables that need to be provided from the user are:

- Initial number of simulation runs: To calculate the initial mean and standard deviation of the sample.
- Desired Confidence Level: It will allow us to calculate the values of the Student's-t distribution ($t_{\alpha/2,n-1}$) and Standard Normal distribution ($z_{\alpha/2}$). In the

dialog window we will only see three options for the confidence level 90 % 95% and 97%

- Maximal error for each measured variable: Allows to stablish real limit around the average for the confidence level. The limit for each variable will be:
    - Throughput Time: 0.2-0.4 Days
    - Mean Tardiness: 0.1-0.2 Days
    - On-Time Delivery: 1-2%

4.2.2 Measured Variables

With the aim of evaluating the performance of the system, the following variables will be evaluated. For each one of them, we will need to compute the number of necessary replications and finally select the biggest one of them. The idea is to achieve a number of replications that give us robust results among all the variables that we want to measure. The validated measures are:

- Average Cycle Time: Represents the average time needed by a wheel set of a specific train since the request is made until the delivery. The already implemented methods provides this value with the name in German and the train name for coding it ."<*TrainCode*>*_DLZ_mittlewert*".

- Mean Tardiness: It collects the difference between the moment when the requested set should be delivered and the moment when is actually delivered. On the following Figure 4-2 we can see an example of the kind of distribution for this variable with the two period moving average added in black. In the simulation model this variable can be found by "*Mittlewert_verspätung_*<*traincode*>".

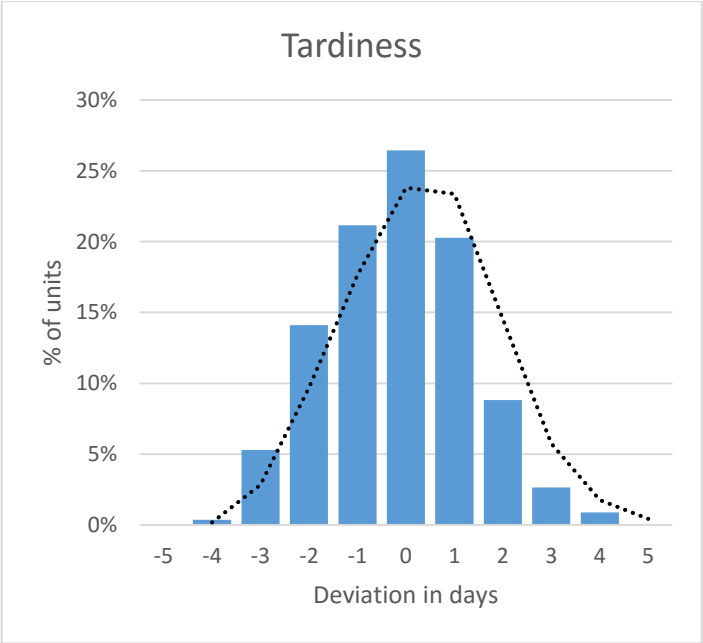


Figure 4-2 Example of Tardiness

- On-Time Delivery: According to the definition founded on [Lea16], the on-time delivery is "*a measure of the process and supply chain efficiency which measures the amount of good or services delivered to customers on time* ". It is a measure of how well the company is meeting their deadlines. In the software, after computing the total time of a determinate part in the system and therefore when it is delivered, the number of parts that are delivered on time is calculated by comparing the time when the set should be delivered with when the set is actually delivered. After classifying the parts, the simulation model now computes the variable On-Time Delivery with the following equation (23):

$$On - Time\ Delivery = \frac{Orders\ delivered\ on\ time}{Total\ orders\ shipped} \tag{23}$$

From each one of this variables we need to calculate different parameters depending on the method used, for instance, when implementing the iterative t-method, we will need the standard deviation, thus the average across replications.

4.3 Output Variables

In the simulation model of the Smart Wheel Set or "*Radsatzwerkstatt*" in German, the sets from two trains are processed, the two train models are VT612 and VT642. For each set, and as we have seen in the previous section, we have several variables that we would like to measure, thus in the end we will have a number of variables that is:

$$N_{Variables} = Number\ of\ wheel\ sets * Number\ of\ measured\ variables \quad (24)$$

Each of this variables will be code as:

```
N_<trainCode>_<MeasuredVariable>
```

However, we want to know a number of replications that guarantee robust output result to all our variables of interest, therefore, after computing each N, we will select the maximum among them. By selecting the maximum we will be choosing a value of n that satisfy the requirements for all the variables. The other output variables will be the same measured variables (cycle-time, on time delivery and mean tardiness). For each of this variables the model will show a plot with the average and the confidence interval for each replication. With this plot we will be able to evaluate if computing one more simulation run its worthy or not. On the Figure 4-3, generated with dummy data, we can appreciate the kind of plots that we will find later on.

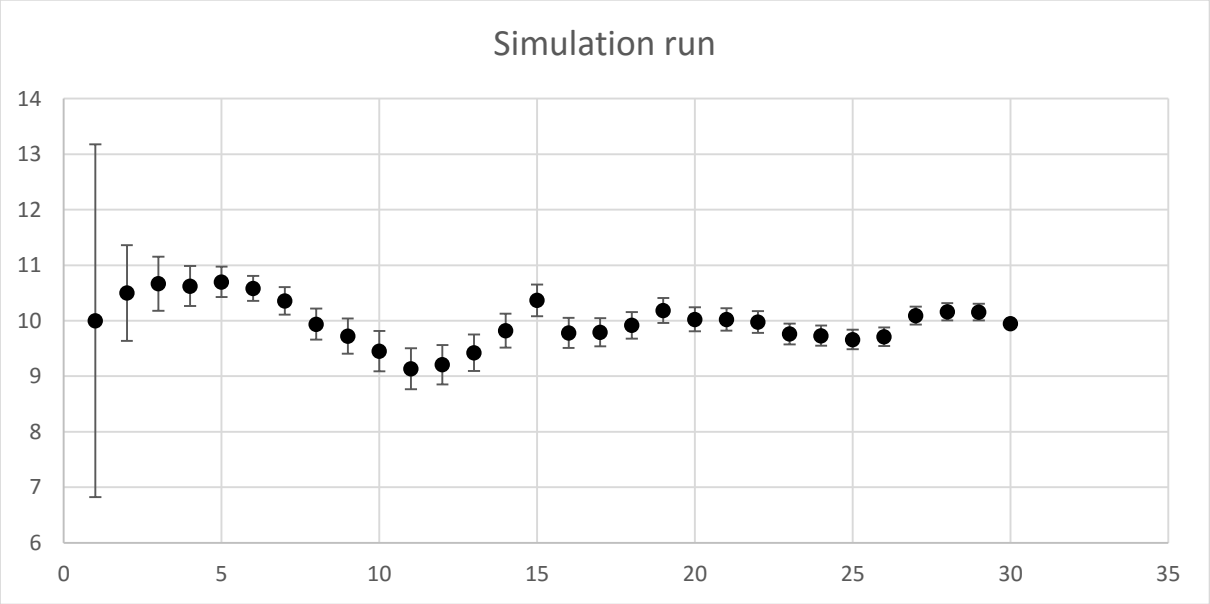


Figure 4-3 Example for output variables

4.4 Developing a T-test

As it was explained in chapter Analysing Simulation Output, the iterative t-test method needs to check on every simulation run if the presented condition is fulfilled. The equation that needs to be checked is:

$$h = t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \leq E \tag{25}$$

To develop the implementation of this method the first thing that should be done is have access to the variables from the equation on each simulation run. We already know form the previous section that E is the limit set by the user for each of the variables, then the variable $t_{\alpha/2,n-1}$ is read from a table that has been previously imported to the simulation model ("*T-tabelle*"). The table can be easily imported from an excel file that allows us to compute the value of the student's t. To read the value of the table, two local variables are defined, "alpha_2", that have the value to be read directly from the table and the "*degrees of freedom*" that contains the number of simulation runs performed minus 1. "Alpha_2" is computed like in the following equation:

$$Alpha_2 = \frac{1-Confidence\ level(\%)}{2} \tag{25}$$

To read from the table in the method called "Inputs", two loops are performed, one to look for the row and another to look for the column. Once that we have the number or name of the row and column, we can easily read the value by the command:

```
T-tabelle [column_number, raw_number];
```

With the other initial number of simulation runs, the user decide the size of the initial sample used to calculate the other variable s (standard deviation) that is computed as follows:

$$s = \sqrt{\frac{\sum_{i=1}^{i=N}(X_1-\bar{X})^2}{n}} \tag{26}$$

In the model, the average of each simulation run for each component is already calculated in some cases, like for the cycle time so we only need to read the values for each one of them from the table "DLZ_Mittlewerte", sum them and calculate the average. All this is done in the method "Inputs" with a loop that reads the values from the above mentioned table. The average for each variable is designated as

"*Average_<variablename>_<traincode>*".Since this method is iterative, the method inputs is evaluated after each simulation run.

In the end to calculate the standard deviation we need to know the differences between the average across replications and each average of each replication. This differences are also computed with a loop and saved on a table file called "Differences".

Finally the method is programmed on a separate object that its run if the iterative-t method is selected on the dialog window. In this other object, called "T-method" the model computes the value of h and compared with the limit E, if h is bigger than E, then it increases the value of the limit of simulation runs (Durchlauf). The system runs until the desired level if achieved. The whole process is shown in Figure 4-4.

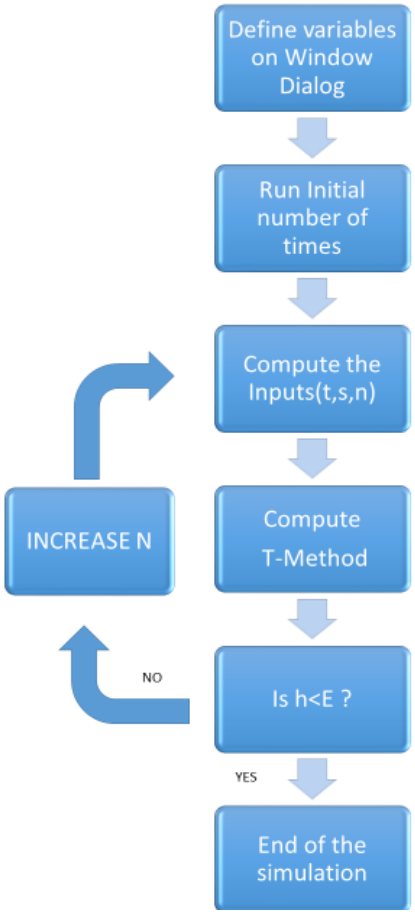


Figure 4-4 Information Flow in iterative t-test method

4.5 Developing a Half-Width Method

One of the main differences of this method with the previous one is that it does not require to iterate to calculate the number of simulation runs. The method, as well as the t-test, require an initial number of simulation runs but since the nature of the iteration of the first method, any number could be valid for the t-test. This is not the case in the half width method, here according to the literature [Ross16], an initial number between 10-20 simulation runs should be set. This number is based on the experience of the author of the book and could be even enough in some cases. The equation that gives us the number of simulation runs is:

$$n \cong n_0 \left(\frac{h_0}{h}\right)^2 \tag{27}$$

Where n_0 is the initial number of simulation runs, h is the limits fixed by the user and h_0 is the half-width that is calculated as in the next equation

$$h_0 = t_{\alpha/2,n_0-1}\frac{s_0}{\sqrt{n_0}} \tag{28}$$

Notice that the value of h_0 or half-width is the same as h in the previous method for the initial number of runs, here is the reason why in the method of the half-width the code and the declared local variables are basically the same.

After computing the number of replications required, the method find the maximum n that validates all the variables al fix the maximum to it. In the Figure 4-5, the information flow for this method is depicted.

Figure 4-5 Information Flow in iterative half-width method

4.6 Developing the ratio-method

The ratio method is somehow similar to the half-width method since it does not iterate. The method is based on the idea that the ratio between the standard deviation and the average remains constant. It is not possible to know before running the simulation model if this condition will be achieved or not, therefore one of the things done in the validation is to measure if this ratio remains constant and in case it is not, declare the method as not valid. Another feature from this method is that the limit is not given by an absolute number but by a % of the mean. The formula that provides the number of simulation runs is the following:

$$n \cong \left(\frac{z_{\alpha/2}}{w} CV\right)^2 \tag{29}$$

Where n is the new limit number of simulation runs, $z_{\alpha/2}$ is the value of the normal table with an α confidence level, that will be read from a table in the model, w is the % of error that the user wants to assume and CV is the ratio at the initial number of simulation runs. This ratio is computed as:

$$CV = \frac{s}{\mu} \tag{30}$$

The variable $z\alpha_{/2}$ is read from a table that has been previously imported to the simulation model ("*T-tabelle*"). The table can be easily imported from an excel file that allows us to compute the value of the z. To read the value of the table, a local variable is defined, "alpha_2" that have the value to be read directly from the table and that is computed like in the following equation:

$$Alpha_2 = \frac{1-Confidence\ level(\%)}{2} \tag{31}$$

Since the values given by the user for the error are designed to be absolute values, this model computes the % of the average by simply dividing the provided error limit by the mean of the variable at the initial number of simulation runs. Finally, to compute the average and standard deviation the model uses the same code as the iterative method. The differences among the different methods rise in the different assumptions of considering the population. While the first method only tries to make simulation runs until the level is reached, this other two methods have the possibility to preview the amount of resources needed to achieve the value. Below (Figure 4-6), the information flow is presented, and as mentioned above, is pretty similar to the half-width method.

Figure 4-6 Information Flow for the Ratio-Method

5 IMPLEMENTATION THE METHOD IN THE SOFTWARE

This section presents the source code and give an explanation for each line or group of lines of the code. It starts with a description of the model and the global variables declared. Then the section starts with a description of how the user defined variables are written, how the values for the t and z tables are computed and read and finally how the methods are implemented in different layers. In the Figure 5-1 the simulation model "*Zents_Simulationsmodell_Radsatzwerkstatt*" where the methods have been implemented is presented. From this model, only some objects will be used, the "DLZ_Mittelwerte", "Mittelwert_Verspätung_612", "Mittelwert_Verspätung_642" tables where the values of the variables that we want to measure are stored. The method "loeschen" that allows to clean the tables after the validation process or every time that the process is initialised and the methods "Produtionsmenge_DLZ_VT612_LR" and "Produtionsmenge_DLZ_VT642_TR" where all the variables for the model as well as the control for the loops is already programmed.

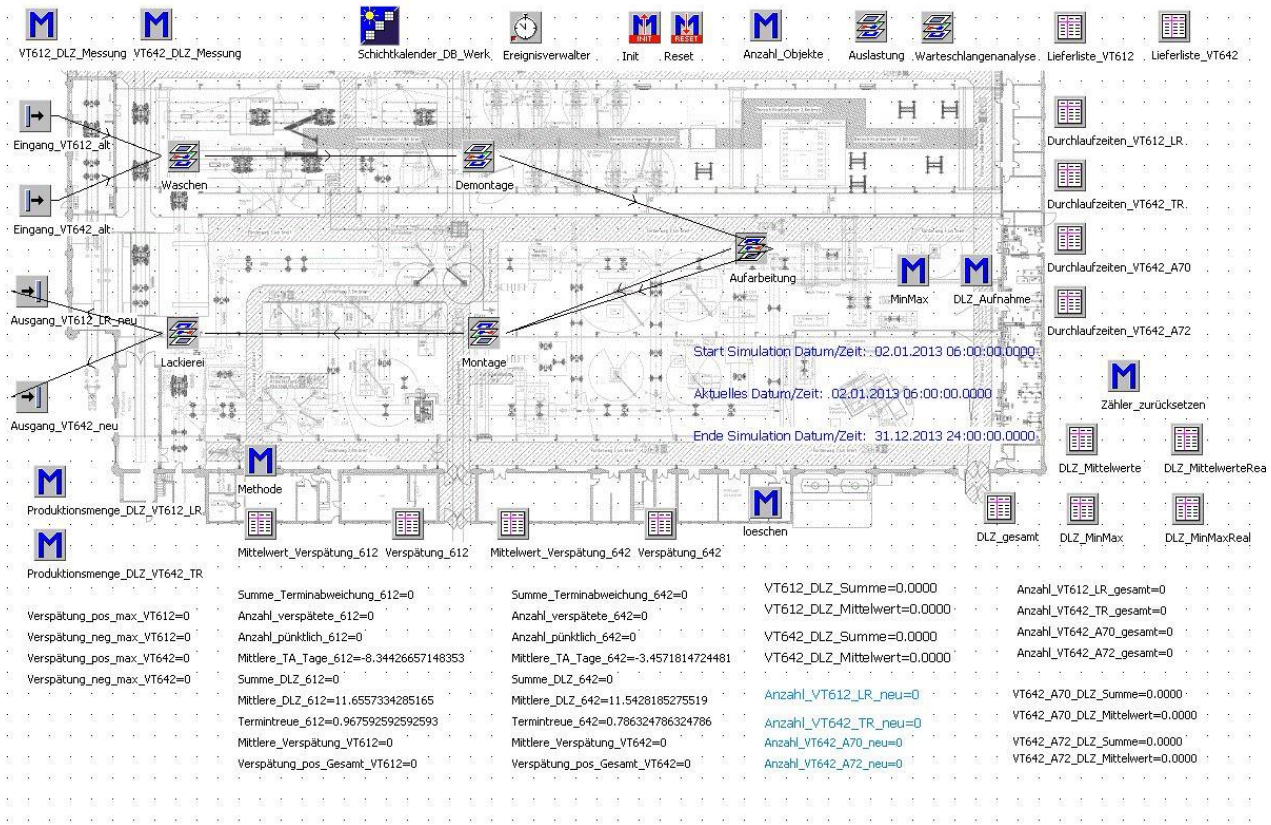


Figure 5-1Simulation Model

In the Figure 5-2 the validation frame is presented. The frame could be divided into 5 sections.

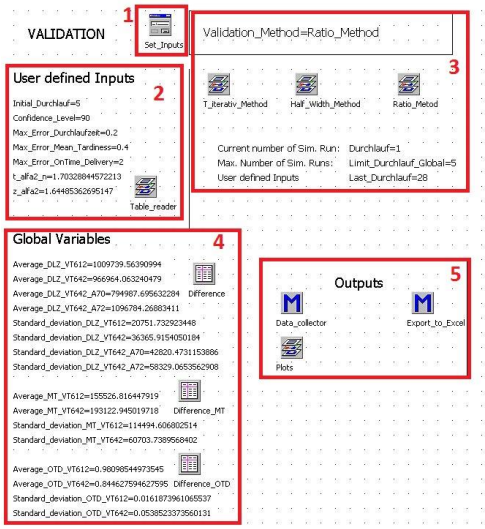


Figure 5-2 Validation Frame

The first one, is the window dialog, used to give the user-defined variables as well as to set to the initialization point the model. The 2nd, shows the variables defined in the window dialog and the values read on the tables that change with the number of simulation run. Both areas can be seen in Figure 5-3.

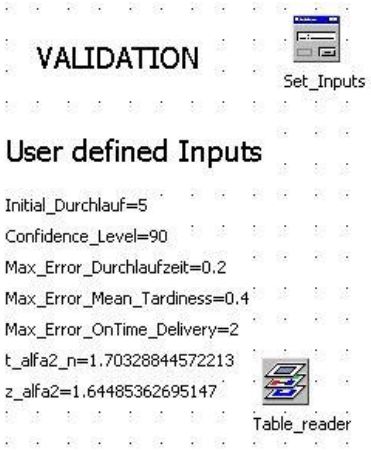


Figure 5-3 Inputs values

The 3rd area (Figure 5-4) contains the current selected validation method, the different methods implemented, the number of the current simulation run and the limit. It also shows the number of simulation runs performed in the last successful validation.

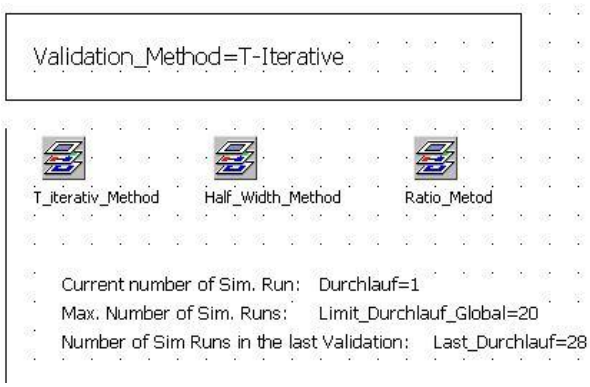


Figure 5-4 Validation Methods

The 4th area (Figure 5-5), includes all the global variables that are used by the three validation methods, an average and standard deviation for each measured variable and train type. It has as well 3 tables where the values for computing the standard deviation are stored.

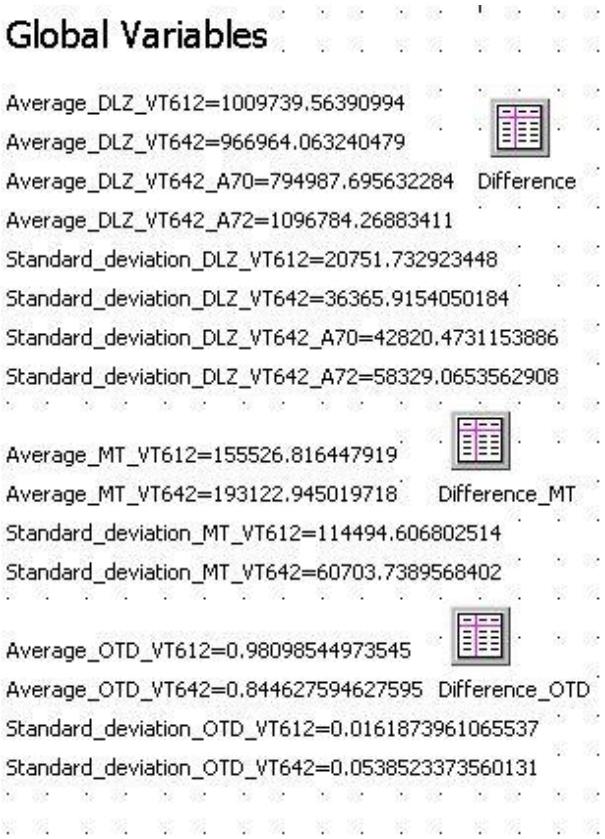


Figure 5-5 Global Variables

Finally in the chart no.5 (Figure 5-6), we could see the outputs of the model, the method "Data_Collector", is in charge of writing the values in tables and computing the normalize values in order to avoid confidentiality troubles. Another option presented in the outputs that is the possibility to export all the data into an excel file.

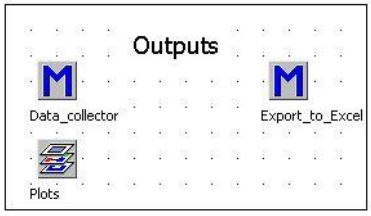


Figure 5-6 Outputs

5.1 Implementation for User-Defined Variables

In order to provide a better user experience and avoid to change manually the input variables, as well as to use it as reset system, a window dialog is implemented. In the section 3.2.4, a brief explanation for it can be found. The window dialog is implemented by adding objects to it and then placing them manually. The Figure 5-7 shows the design window. The options shown hereunder, are the "Statische Texfeld" that allows

the user to write text without interacting with it, the “Textfeld” that allows to provide any type of information in a blank and the “Dropdownlistenfeld” that creates a list of options or values where only one can be selected. By pressing the button “Dialog bearbeiten”, the model shows a preview of the dialog window and allows to move elements freely.

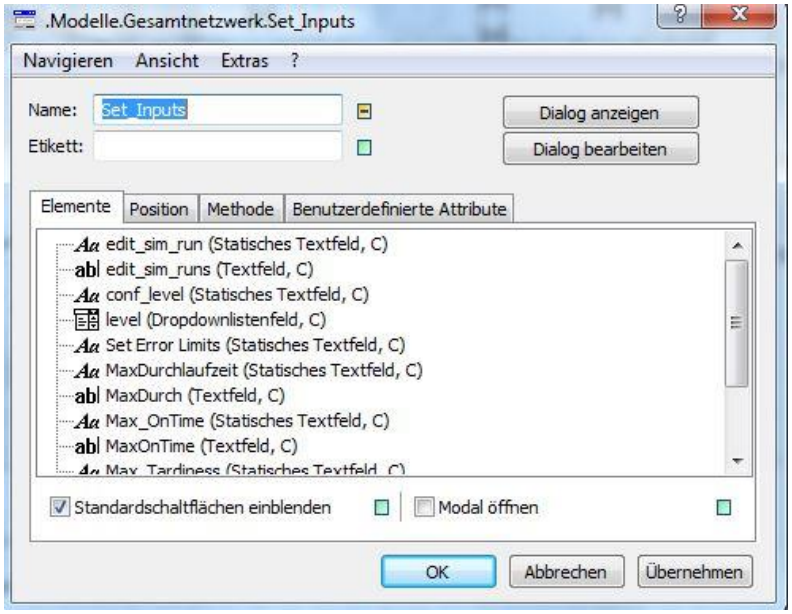


Figure 5-7 Window Dialog (Edit window)

However, the implementation of the window should also include the reset of all variables and tables in order to provide a clean start for the validation. The window dialog object provides a Call.Back method that it is executed when any of the actions are carried out. The code for this method is presented in Figure 5-8.

```
(action : string)
is
do
    inspect action -- check the accion done
    when "Open" then
        -- TODO: add code for the "Open" action here

    when "Apply" then

        -- read the values of the "Textfelds"
        Initial_Durchlauf:=
        str_to_num(Set_Inputs.holeWert("edit_sim_runs"));
        Confidence_Level:=
        str_to_num(Set_Inputs.holeWert("level"));
        Max_Error_Durchlaufzeit:=
        str_to_num(Set_Inputs.holeWert("MaxDurch"));
        Max_Error_Mean_Tardiness:=
        str_to_num(Set_Inputs.holeWert("Maxtardiness"));
        Max_Error_OnTime_Delivery:=
        str_to_num(Set_Inputs.holeWert("MaxOnTime"));

        Validation_Method:= Set_Inputs.holewert("Selection");

        -- Clear tables
        DLZ_Mittelwerte.loeschen();
        DLZ_MittelwerteReal.loeschen();
        Plots.Data_Gathering.loeschen();
        Plots.Data_Gathering_Percent.loeschen();
        loeschen;
        -- Set the initial values for the
        Durchlauf:=1;
        Limit_Durchlauf_Global:=Initial_Durchlauf;
        -- Reset the timer
        ereignisverwalter.ruecksetzen;

          -- Set Initial Replication limits
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_DLZ_VT612:= 1;
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_DLZ_VT642:= 1;
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_DLZ_VT642_A70:=1;
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_DLZ_VT642_A72:=1;
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_MT_VT612:=1;
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_MT_VT642:=1;
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_OTD_VT612:=1;
          .modelle.gesamtnetzwerk.T_iterativ_Method.TI_n_replications_OTD_VT642:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_DLZ_VT612:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_DLZ_VT642:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_DLZ_VT642_A70:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_DLZ_VT642_A72:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_MT_VT612:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_MT_VT642:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_OTD_VT612:=1;
          .modelle.gesamtnetzwerk.Half_Width_Method.HW_n_replications_OTD_VT642:=1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_DLZ_VT612:= 1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_DLZ_VT642:= 1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_DLZ_VT642_A70:= 1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_DLZ_VT642_A72:= 1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_MT_VT612:= 1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_MT_VT642:= 1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_OTD_VT612:=1;
          .modelle.gesamtnetzwerk.ratio_Metod.RM_n_replications_OTD_VT642:=1;
    when "Close" then
        -- TODO: add code for the "Close" action here
    end;
end;
```

Figure 5-8 Call.Back Method

In the code above, another method, called "*loeschen*" is called, this method was already implemented in the model and helps to clean the table. The method is shown below in Figure 5-9.

```
is
do
	DLZ_gesamt.loeschen();
	Verspätung_612.loeschen();
	Verspätung_642.loeschen();
	DLZ_MinMax.loeschen();
	DLZ_MinMaxReal.loeschen();
	Mittelwert_Verspätung_612.loeschen();
	Mittelwert_Verspätung_642.loeschen();
end;
```

Figure 5-9 Loeschen Method

This window dialog sets the values written in the box and also restart the model every time that "apply" or "Übernehmen" in the German version is pushed. Notice that the confidence levels are limited to 3 common options 90%-95%-97% as well as the validation methods as shown in the Figure 5-10.

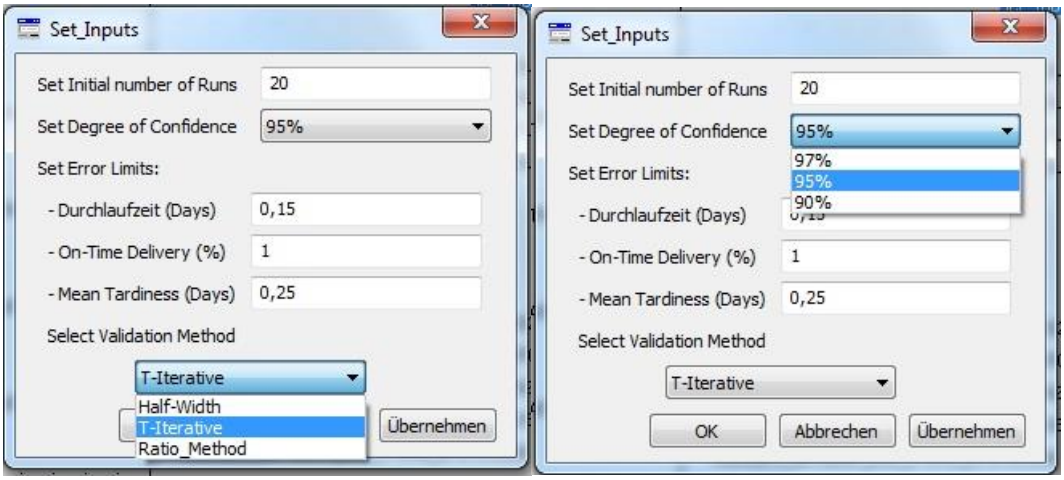


Figure 5-10 Settings for the dialog window

The final result can be seen in Figure 5-11

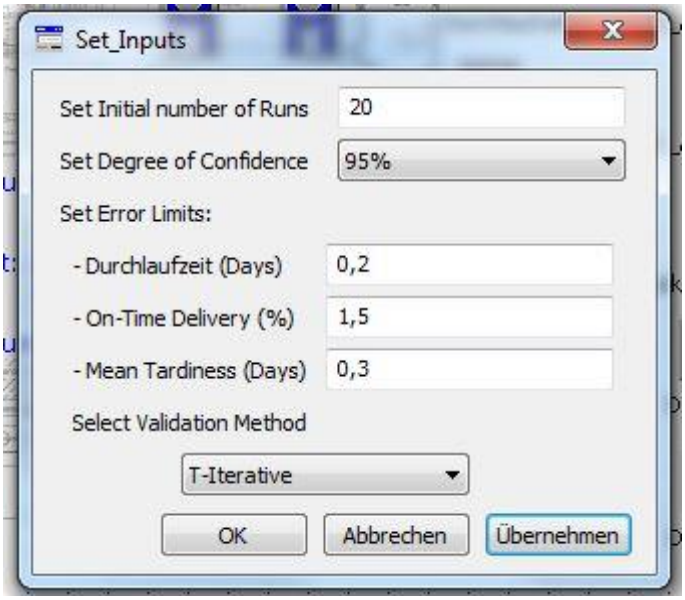


Figure 5-11 Dialog Window

In addition to the dialog window, the other inputs variables, those read from the T and Z tables, are needed but before reading them with a method. The two tables have been computed in Excel. To compute the T table, excel provides a function T.INV(probability;deg_freedom) that allows to have any value given the value of alpha and the degree of freedom. In Figure 5-12 we can see the probability in the first raw and the different degrees of freedom in the first column. The table is computed until 300 but notice that those values are the same as the values from the standard normal distribution.

	A	B	C	D	E	F	G	H	I	J
1				T DISTRIBUTION						
2		0,25	0,2	0,15	0,1	0,05	0,025	0,0125	0,01	0,005
3	1	1	1,376382	1,962611	3,077684	6,313752	12,7062	25,4517	31,82052	63,65674
4	2	0,816496581	1,06066	1,386207	1,885618	2,919986	4,302653	6,205347	6,964557	9,924843
5	3	0,764892328	0,978472	1,249778	1,637744	2,353363	3,182446	4,176535	4,540703	5,840909
6	4	0,740697084	0,940965	1,189567	1,533206	2,131847	2,776445	3,495406	3,746947	4,604095
7	5	0,726686844	0,919544	1,155767	1,475884	2,015048	2,570582	3,163381	3,36493	4,032143
8	6	0,717558196	0,905703	1,134157	1,439756	1,94318	2,446912	2,968687	3,142668	3,707428
9	7	0,711141778	0,89603	1,119159	1,414924	1,894579	2,364624	2,841244	2,997952	3,499483
10	8	0,706386613	0,88889	1,108145	1,396815	1,859548	2,306004	2,751524	2,896459	3,355387
11	9	0,702722147	0,883404	1,099716	1,383029	1,833113	2,262157	2,685011	2,821438	3,249836
12	10	0,699812061	0,879058	1,093058	1,372184	1,812461	2,228139	2,633767	2,763769	3,169273
13	11	0,697445328	0,87553	1,087666	1,36343	1,795885	2,200985	2,593093	2,718079	3,105807

Figure 5-12 T-table

The second table, called Z-table it is even easier since it does not depend on the degree of freedom. The function that gives us the value is NORM.S.INV(probability). This table is shown in the Figure 5-13 below:

		Z DISTRIBUTION								
2		0,25	0,2	0,15	0,1	0,05	0,025	0,0125	0,01	0,005
3	z(alpha)	0,67	0,84	1,04	1,28	1,64	1,96	2,24	2,33	2,58

Figure 5-13 Z-Table

Once that we have both tables, the model runs the method called “Table_Reading” (that is inside the frame “Table_reader”) every simulation loop and computes the values for T and Z. It is important to notice that in this method, we need to put the path where the computed table are in the document to import them, however, once that they are imported. It’s unnecessary to do it again, therefore this lines are commented. The code it is self-explained in the Figure 5-14, as a result, no longer explanations are needed.

```
/*This method is run on every single simulation run in order to have the values of t for the t-iterative method
and of z for the ratio-method*/

is
    -- dummy variables to read the columns and rows
    i:integer;
    j:integer;
    -- variables to store the values from the columns of tables T and Z
    column_value_t:integer;
    column_value_z:integer;
    row_value_t:integer;
    degrees_freedom:integer;
    alfa_2:real;

do
    ------------ IMPORT EXCEL Tables to Plan SImulation ------------------
    /*t_tabelle.leseExcelDatei("Z:\ModellCopy\T-TEST.xlsx");
    z_tabelle.leseExcelDatei("Z:\ModellCopy\z-table.xlsx");*/
```

```
----------- LOOPS to FIND THE VALUES OF THE COLUMN AND RAW ON EACH TABLE ----------
alfa_2:= (100-.modelle.gesamtnetzwerk.Confidence_Level)/200;
degrees_freedom:= .modelle.gesamtnetzwerk.durchlauf-1;
-- column search T table
i:=1;
j:=2;

while str_to_num(T_tabelle[i,j]) /= alfa_2 loop
    column_value_t:=i+1;
    i:=i+1;
    --print column_value_t;
end;

-- column search Z table
i:=1;
j:=2;
while str_to_num(Z_tabelle[i,j]) /= alfa_2 loop
    column_value_z:=i+1;
    i:=i+1;
    --print  column_value_z;
end;

-- row search table T
i:=1;
j:=3;

row_value_t:=j;

while str_to_num(T_tabelle[i,j]) /= Degrees_freedom loop
    j:=j+1;
    row_value_t:=j;
    --print  row_value_t;
end;

------------ READ THE STANDARD VALUES FROM THE IMPORTED TABLES----------------

if .modelle.gesamtnetzwerk.durchlauf==1 then
    .modelle.gesamtnetzwerk.t_alfa2_n:=0;
else
    .modelle.gesamtnetzwerk.t_alfa2_n:= str_to_num(T_tabelle[column_value_t,row_value_t]);
end;

.modelle.gesamtnetzwerk.z_alfa2:= str_to_num(Z_tabelle[column_value_z,3]);
end;
```

Figure 5-14 Table-Reader Method

The values of imported tables from excel are saved in two tables in Plant Simulation called “*t_tabelle*” and “*z_tabelle*” that can be found in the frame “Table_reader”

5.2 Implementation of Validation Methods

In the implementation of the validation algorithm, the codes for the different methods are detailed explained, however, the computation of the global variables will be explained only in the T-iterative method and it will be common for the others. As it has been explained before, the idea is to perform simulations until the limit is achieved.

This limit is shown in the variable "*Limit_Durchlauf_Global*" (Figure 5-4) and it is controlled by the selected method in the dialog window. To perform this control, the method mentioned before *"Produktionsmenge_DLZ_VT612_LR" and* "*Produktionsmenge_DLZ_VT612_LR"* were edited by adding this code (FIGURE) at the end of it. Notice that the code is a part of the already implemented code from the model and therefore it does not follow the standard structure is-do-end. This method is computed after every simulation run.

```
-- After computing all the variables the model read the tables
.modelle.gesamtnetzwerk.Table_reader.table_Reading; --path to the table reader method
-- Control of the methods and they are run only after the 2 sim.run
if Validation_Method == "T-Iterative" AND durchlauf>1 then
    T_iterativ_Method.T_Iterative;
end;

if Validation_Method == "Half-Width" AND durchlauf>1 then
    Half_Width_Method.half_Width;
end;

if Validation_Method == "Ratio_Method" AND durchlauf>1 then
    Ratio_Metod.Ratio_Method;
end;


Durchlauf := Durchlauf +1;      -- increase the variable that count the current Sim. run
Ereignisverwalter.ruecksetzen;  --Reset the timer
Ereignisverwalter.start;        --Start the timer
.Modelle.Gesamtnetzwerk.Auslastung.Addieren;

if Durchlauf = (Limit_Durchlauf_Global+1) then -- if the limit is reached
    Ereignisverwalter.stop;             -- stop the simulation
    Durchlauf :=1;                      -- set the current number to 1
    Last_Durchlauf:= Limit_Durchlauf_Global;   --Set the last sucessful number of sim runs
    Limit_Durchlauf_Global:= Initial_Durchlauf; --reset the limit to the initial number of sim runs
end;
```

Figure 5-15 Control of the Selected method and the Limit of Sim. Runs

5.2.1 T-Iterative

In the frame T-Iterative, we can find two objects, the first one are the variables output of the method and the other is the method itself, both objects can be seen in Figure 5-16.

Figure 5-16 Objects in the frame T-Iterative

The code will be explained hereunder in different figures to make it easier to understand. First of all the declaration of the local variables is shown and explained in the Figure 5-17.

```
is
    i:integer; --index to evaluate values


    -- Durchlaufzeit(DLZ)_<train_type> local variables to measure average and standard deviation
    Gesamt_DLZ_VT612:time;
    Gesamt_DLZ_VT642:time;
    Gesamt_DLZ_VT642_A70:time;
    Gesamt_DLZ_VT642_A72:time;
    Sample_variance_DLZ_VT612: real;
    Sample_variance_DLZ_VT642: real;
    Sample_variance_DLZ_VT642_A70: real;
    Sample_variance_DLZ_VT642_A72: real;

    -- On-Time Delivery(OTD)_<train_type> local variables to measure average and standard deviation variables
    Gesamt_OTD_VT612:time;
    Gesamt_OTD_VT642:time;
    Sample_variance_OTD_VT612: real;
    Sample_variance_OTD_VT642: real;

    -- Mean Tardiness delivery(MT)_<train_type> local variables to measure average and standard deviation
    Gesamt_MT_VT612:real;
    Gesamt_MT_VT642:real;
    Sample_variance_MT_VT612:real;
    Sample_variance_MT_VT642:real;

    -- DLZ limits
    h_DLZ_VT612:real;
    h_DLZ_VT642:real;
    h_DLZ_VT642_A70:real;
    h_DLZ_VT642_A72:real;
    -- OTD Limits
    h_OTD_VT612:real;
    h_OTD_VT642:real;
    -- MT Limits
    h_MT_VT612:real;
    h_MT_VT642:real;

    -- Variable to compute the limit of DLZ(is given in Days)
    UserDefined_limit_DLZ:real;
    -- variable to compute the limit of OTD (given in %)
    UserDefined_limit_OTD:real;
    -- Variable to compute the limit of MT (is given in Days)
    UserDefined_limit_MT:real;
```

Figure 5-17 Local Variables T-Iterative

The next image shows the computation of the average, Figure 5-18 and the transformation of the errors given by the user that are given in Days or %. All the values are in seconds (for DLZ and MT).

```
do
    -- Make necessary transformations to have the values able to work with them
    UserDefined_limit_DLZ:=.modelle.gesamtnetzwerk.Max_Error_Durchlaufzeit*86400; -- Transform the value into seconds
    UserDefined_limit_MT:=.modelle.gesamtnetzwerk.Max_Error_Mean_Tardiness*86400; -- Transform the value into seconds
    UserDefined_limit_OTD:=.modelle.gesamtnetzwerk.Max_Error_OnTime_Delivery/100; -- The values are a ratio, thus we should convert
                                                                                  -- the % provided by user

    ---------- AVERAGE ACROSS REPLICATIONS ----------
    -- Initialize the loop to sum all the averages values of each replication
    i:=1;
    repeat
        Gesamt_DLZ_VT612:= .modelle.gesamtnetzwerk.DLZ_Mittelwerte[2,i] + Gesamt_DLZ_VT612;  --Calcualates the cumulative value of time for each type
        Gesamt_DLZ_VT642:= .modelle.gesamtnetzwerk.DLZ_Mittelwerte[3,i]+ Gesamt_DLZ_VT642;
        Gesamt_DLZ_VT642_A70:=.modelle.gesamtnetzwerk.DLZ_Mittelwerte[4,i]+ Gesamt_DLZ_VT642_A70;
        Gesamt_DLZ_VT642_A72:=.modelle.gesamtnetzwerk.DLZ_Mittelwerte[5,i]+ Gesamt_DLZ_VT642_A72;
        Gesamt_MT_VT612:=.modelle.gesamtnetzwerk.Mittelwert_Verspätung_612[7,i]*86400+Gesamt_MT_VT612; -- this values are given in days in the model
        Gesamt_MT_VT642:=.modelle.gesamtnetzwerk.Mittelwert_Verspätung_642[7,i]*86400+Gesamt_MT_VT642; -- and we want them in seconds, thus *86400
        Gesamt_OTD_VT612:=Gesamt_OTD_VT612+.modelle.gesamtnetzwerk.Mittelwert_Verspätung_612[6,i];
        Gesamt_OTD_VT642:=Gesamt_OTD_VT642+.modelle.gesamtnetzwerk.Mittelwert_Verspätung_642 [6,i];
        i:=i+1;
    until i> .modelle.gesamtnetzwerk.durchlauf; -- the limit is the number of current sim. runs

    -- Calculate the averages of each varaible
    .modelle.gesamtnetzwerk.Average_DLZ_VT612:= Gesamt_DLZ_VT612 / .modelle.gesamtnetzwerk.Durchlauf;
    .modelle.gesamtnetzwerk.Average_DLZ_VT642:= Gesamt_DLZ_VT642 / .modelle.gesamtnetzwerk.Durchlauf;
    .modelle.gesamtnetzwerk.Average_DLZ_VT642_A70:= Gesamt_DLZ_VT642_A70/ .modelle.gesamtnetzwerk.Durchlauf;
    .modelle.gesamtnetzwerk.Average_DLZ_VT642_A72:= Gesamt_DLZ_VT642_A72/ .modelle.gesamtnetzwerk.Durchlauf;
    .modelle.gesamtnetzwerk.Average_MT_VT612:= Gesamt_MT_VT612 / .modelle.gesamtnetzwerk.durchlauf;
    .modelle.gesamtnetzwerk.Average_MT_VT642:= Gesamt_MT_VT642 / .modelle.gesamtnetzwerk.durchlauf;
    .modelle.gesamtnetzwerk.Average_OTD_VT612:= Gesamt_OTD_VT612 / .modelle.gesamtnetzwerk.durchlauf;
    .modelle.gesamtnetzwerk.Average_OTD_VT642:= Gesamt_OTD_VT642 / .modelle.gesamtnetzwerk.durchlauf;
```

Figure 5-18 Average and error computation

The next Figure 5-19 shows the computation of the standard deviation:

```
    ---------- STANDARD DEVIATION ----------------
    -- Initialice the loop
    i:=1;
    -- cleans the table, after each sim run, the average change, therefore the whole table needs to be
    -- recomputed.
    .modelle.gesamtnetzwerk.Difference.loeschen();
    .modelle.gesamtnetzwerk.Difference_MT.loeschen();
    .modelle.gesamtnetzwerk.Difference_OTD.loeschen();
    repeat
        -- Differences between the averages and the value of each run. All the values are saved on tables
        -- DLZ
        .modelle.gesamtnetzwerk.Difference[1,i]:= i;
        .modelle.gesamtnetzwerk.Difference[2,i]:= time_to_num(.modelle.gesamtnetzwerk.DLZ_Mittelwerte[2,i]-.modelle.gesamtnetzwerk.Average_DLZ_VT612);
        .modelle.gesamtnetzwerk.Difference[3,i]:= time_to_num(.modelle.gesamtnetzwerk.DLZ_Mittelwerte[3,i]-.modelle.gesamtnetzwerk.average_DLZ_VT642);
        .modelle.gesamtnetzwerk.Difference[4,i]:= time_to_num(.modelle.gesamtnetzwerk.DLZ_Mittelwerte[4,i]-.modelle.gesamtnetzwerk.average_DLZ_VT642_A70);
        .modelle.gesamtnetzwerk.Difference[5,i]:= time_to_num(.modelle.gesamtnetzwerk.DLZ_Mittelwerte[5,i]-.modelle.gesamtnetzwerk.average_DLZ_VT642_A72);
        -- MT
        .modelle.gesamtnetzwerk.Difference_MT[1,i]:=i;
        .modelle.gesamtnetzwerk.Difference_MT[2,i]:= (.modelle.gesamtnetzwerk.Mittelwert_Verspätung_612[7,i]*86400-.modelle.gesamtnetzwerk.Average_MT_VT612);
        .modelle.gesamtnetzwerk.Difference_MT[3,i]:= (.modelle.gesamtnetzwerk.Mittelwert_Verspätung_642[7,i]*86400-.modelle.gesamtnetzwerk.Average_MT_VT642);
        -- OTD
        .modelle.gesamtnetzwerk.Difference_OTD[1,i]:=i;
        .modelle.gesamtnetzwerk.Difference_OTD[2,i]:= (.modelle.gesamtnetzwerk.Mittelwert_Verspätung_612[6,i]-.modelle.gesamtnetzwerk.Average_OTD_VT612);
        .modelle.gesamtnetzwerk.Difference_OTD[3,i]:= (.modelle.gesamtnetzwerk.Mittelwert_Verspätung_642[6,i]-.modelle.gesamtnetzwerk.Average_OTD_VT642);
        ---------Pow(differences,2)
        -- DLZ
        .modelle.gesamtnetzwerk.Difference[6,i]:= pow(.modelle.gesamtnetzwerk.Difference[2,i],2);
        .modelle.gesamtnetzwerk.Difference[7,i]:= pow(.modelle.gesamtnetzwerk.Difference[3,i],2);
        .modelle.gesamtnetzwerk.Difference[8,i]:= pow(.modelle.gesamtnetzwerk.Difference[4,i],2);
        .modelle.gesamtnetzwerk.Difference[9,i]:= pow(.modelle.gesamtnetzwerk.Difference[5,i],2);
        -- MT
        .modelle.gesamtnetzwerk.Difference_MT[4,i]:= pow (.modelle.gesamtnetzwerk.Difference_MT[2,i],2);
        .modelle.gesamtnetzwerk.Difference_MT[5,i]:= pow (.modelle.gesamtnetzwerk.Difference_MT[3,i],2);
        -- OTD
        .modelle.gesamtnetzwerk.Difference_OTD[4,i]:= pow (.modelle.gesamtnetzwerk.Difference_OTD[2,i],2);
        .modelle.gesamtnetzwerk.Difference_OTD[5,i]:= pow (.modelle.gesamtnetzwerk.Difference_OTD[3,i],2);
        -------- Sum of pow differences
        -- DLZ
        .modelle.gesamtnetzwerk.Difference[10,1]:=.modelle.gesamtnetzwerk.Difference[6,i]+.modelle.gesamtnetzwerk.Difference[10,1];
        .modelle.gesamtnetzwerk.Difference[11,1]:=.modelle.gesamtnetzwerk.Difference[7,i]+.modelle.gesamtnetzwerk.Difference[11,1];
        .modelle.gesamtnetzwerk.Difference[12,1]:=.modelle.gesamtnetzwerk.Difference[8,i]+.modelle.gesamtnetzwerk.Difference[12,1];
        .modelle.gesamtnetzwerk.Difference[13,1]:=.modelle.gesamtnetzwerk.Difference[9,i]+.modelle.gesamtnetzwerk.Difference[13,1];
        --MT
        .modelle.gesamtnetzwerk.Difference_MT[6,1]:=.modelle.gesamtnetzwerk.Difference_MT[4,i]+.modelle.gesamtnetzwerk.Difference_MT[6,1];
        .modelle.gesamtnetzwerk.Difference_MT[7,1]:=.modelle.gesamtnetzwerk.Difference_MT[5,i]+.modelle.gesamtnetzwerk.Difference_MT[7,1];
        --OTD
        .modelle.gesamtnetzwerk.Difference_OTD[6,1]:=.modelle.gesamtnetzwerk.Difference_OTD[4,i]+.modelle.gesamtnetzwerk.Difference_OTD[6,1];
        .modelle.gesamtnetzwerk.Difference_OTD[7,1]:=.modelle.gesamtnetzwerk.Difference_OTD[5,i]+.modelle.gesamtnetzwerk.Difference_OTD[7,1];
        i:= i+1;
    until i>= .modelle.gesamtnetzwerk.durchlauf;
```

```
-- Sample Variance Computation for each variable Sum of Pow(differences,2)/n
--DLZ
sample_variance_DLZ_VT612:= .modelle.gesamtnetzwerk.Difference[10,1] / .modelle.gesamtnetzwerk.durchlauf;
sample_variance_DLZ_VT642:= .modelle.gesamtnetzwerk.Difference[11,1] / .modelle.gesamtnetzwerk.durchlauf;
sample_variance_DLZ_VT642_A70:= .modelle.gesamtnetzwerk.Difference[12,1] / .modelle.gesamtnetzwerk.durchlauf;
sample_variance_DLZ_VT642_A72:= .modelle.gesamtnetzwerk.Difference[13,1] / .modelle.gesamtnetzwerk.durchlauf;
--MT
sample_variance_MT_VT612:= .modelle.gesamtnetzwerk.Difference_MT[6,1]/.modelle.gesamtnetzwerk.durchlauf;
sample_variance_MT_VT642:= .modelle.gesamtnetzwerk.Difference_MT[7,1]/.modelle.gesamtnetzwerk.durchlauf;
--OTD
sample_variance_OTD_VT612:= .modelle.gesamtnetzwerk.Difference_OTD[6,1]/.modelle.gesamtnetzwerk.durchlauf;
sample_variance_OTD_VT642:= .modelle.gesamtnetzwerk.Difference_OTD[7,1]/.modelle.gesamtnetzwerk.durchlauf;

-- Standard Deviation Computation
--DLZ
.modelle.gesamtnetzwerk.Standard_deviation_DLZ_VT612:= sqrt(sample_variance_DLZ_VT612);
.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642:= sqrt(sample_variance_DLZ_VT642);
.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A70:= sqrt(sample_variance_DLZ_VT642_A70);
.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A72:= sqrt(sample_variance_DLZ_VT642_A72);
--MT
.modelle.gesamtnetzwerk.Standard_deviation_MT_VT612:= sqrt(sample_variance_MT_VT612);
.modelle.gesamtnetzwerk.Standard_deviation_MT_VT642:= sqrt(sample_variance_MT_VT642);
--OTD
.modelle.gesamtnetzwerk.Standard_deviation_OTD_VT612:= sqrt(sample_variance_OTD_VT612);
.modelle.gesamtnetzwerk.Standard_deviation_OTD_VT642:= sqrt(sample_variance_OTD_VT642);
```

Figure 5-19 Computation of the standard deviation

Now the method calculate the limit for each one of the variables before iterating. Since the standard deviation is the square root of the sample variance and the number of simulation runs is increasing as well, the limit will decrease with a logarithmic style function. In the Figure 5-20, the computation can be shown as well as some commented lines that were used to check the proper function of the method. At the end, the last line is to call the method to copy the variables in tables and make the plots that will be explained later on.

```
-- Calculate the limits before iterating
--DLZ
h_DLZ_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT612/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h_DLZ_VT642:=.modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h_DLZ_VT642_A70:=.modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A70/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h_DLZ_VT642_A72:=.modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A72/sqrt(.modelle.gesamtnetzwerk.durchlauf);

--MT
h_MT_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_MT_VT612/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h_MT_VT642:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_MT_VT642/sqrt(.modelle.gesamtnetzwerk.durchlauf);

--OTD
h_OTD_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_OTD_VT612/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h_OTD_VT642:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_OTD_VT642/sqrt(.modelle.gesamtnetzwerk.durchlauf);

-- Print in Console to Check DLZ
/*
print h_DLZ_VT612;
print h_dlZ_VT642;
print h_dlZ_VT642_A70;
print h_DLZ_VT642_A72;
print UserDefined_limit_MTM;
*/
-- Print in Console to Check MT
/*
print h_MT_VT612;
print h_MT_VT642;
print UserDefined_limit_MT;
*/
-- Print in Console to Check OTD
/*
print h_OTD_VT612;
print h_OTD_VT642;
print UserDefined_limit_OTD;
*/

-- Copy the values and compute the plots after each iteration
.modelle.gesamtnetzwerk.Data_collector;
```

Figure 5-20 Computation of the limits

The next step made by the system is to check if the limits defined by the user are overtake or not. If they are overtake, then it increases the number of simulation runs. This code can be seen in Figure 5-21.

```
-- Check the computed limits with the user defined and increase the number of sim runs if the limit is not achieved
-- by the initial sim runs.If one of the conditions if fulfilled, the model increase the limit of simulation runs.

if h_dlz_VT612> userDefined_limit_DLZ then
        TI_n_replications_DLZ_VT612:=TI_n_replications_DLZ_VT612+1;
end;

if h_dlz_VT642> userDefined_limit_DLZ then
        TI_n_replications_DLZ_VT642:=TI_n_replications_DLZ_VT642+1;
end;

if h_dlz_VT642_A70> userDefined_limit_DLZ then
    TI_n_replications_DLZ_VT642_A70:=TI_n_replications_DLZ_VT642_A70+1;
end;

if h_dlz_VT642_A72> userDefined_limit_DLZ then
    TI_n_replications_DLZ_VT642_A72:=TI_n_replications_DLZ_VT642_A72+1;
end;

if h_MT_VT612> userDefined_limit_MT then
    TI_n_replications_MT_VT612:=TI_n_replications_MT_VT612+1;
end;

if h_mt_VT642> userDefined_limit_MT then
    TI_n_replications_MT_VT642:=TI_n_replications_MT_VT642+1;
end;

if h_OTD_VT612> userDefined_limit_OTD then
    TI_n_replications_OTD_VT612:=TI_n_replications_OTD_VT612+1;
end;

if h_OTD_VT642> userDefined_limit_OTD then
    TI_n_replications_OTD_VT642:=TI_n_replications_OTD_VT642+1;
end;
```

Figure 5-21 Increasing number of replications

Finally the method check all the variables by comparing ones against each other. First the method gives the variable "*Limit_Durchlauf_Global*" the value of "*TI_n_replications_DLZ_VT612*" then then proceeds to compare this variable with the next one. If the "*Limit_Durchlauf_Global*" is smaller, the limit is updated with the checked variable. Otherwise, the method keeps running to check the next one. The code is shown in Figure 5-22.

```
-- Search for the maximum of N of all the variables
    if .modelle.gesamtnetzwerk.durchlauf >= .modelle.gesamtnetzwerk.limit_Durchlauf_Global then
        .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:= TI_n_replications_DLZ_VT612;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<TI_n_replications_DLZ_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=TI_n_replications_DLZ_VT642;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<TI_n_replications_DLZ_VT642_A70 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=TI_n_replications_DLZ_VT642_A70;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<TI_n_replications_DLZ_VT642_A72 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=TI_n_replications_DLZ_VT642_A72;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<TI_n_replications_MT_VT612 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=TI_n_replications_MT_VT612;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<TI_n_replications_MT_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=TI_n_replications_MT_VT642;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<TI_n_replications_OTD_VT612 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=TI_n_replications_OTD_VT612;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<TI_n_replications_OTD_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=TI_n_replications_OTD_VT642;
        end;
    end;
end;
```

Figure 5-22 Search for the maximum of all the variables in the T-Iterative Method

5.2.2 Half-Width

The method of the half-width it is aimed to provide a number of simulation runs that is close to achieve the level of confidence desired under the assumed errors, however this that now mean that the conditions are 100% fulfilled. It has a great dependence of the initial values thus, a large number of initial simulation runs gives it a better performance. The implementation of the method done in the frame "*Half-Width*" shows the global variables HW_n_replications_<variablename>_<traincode> and the method where the code is programmed (Figure 5-23)

Figure 5-23 Frame of the half-width method

As it has been explained before, the model computes should only compute the average and standard deviation when the simulation runs reach the defined initial value of simulation runs. However, and in order to have the plots and the values of the errors for each simulation run, I have decided to create a code that is run on every simulation run. Hereunder, the code is presented, notice that the code is mainly the same as for the t-iterative and therefore is not commented. Only the parts that are different are explained. First the local variables are the same, but the limits, that are only computed once when “Durchlauf = Initial_Durchlauf” this can be shown in the figure below (Figure 5-24):

```
-- DLZ limits
h0_DLZ_VT612:real;
h0_DLZ_VT642:real;
h0_DLZ_VT642_A70:real;
h0_DLZ_VT642_A72:real;
-- OTD Limits
h0_OTD_VT612:real;
h0_OTD_VT642:real;
-- MT Limits
h0_MT_VT612:real;
h0_MT_VT642:real;
```

Figure 5-24 Half-Width Error

The computation of the average and standard deviation is exactly the same as for the previous method. It can be seen in Figure 5-18 and Figure 5-19. After having all the inputs the method computes the limits h_0 for each variable and collect the data for the plots this is made in Figure 5-25

```
-- Calculates the limits for each simulation run
--DLZ
h0_DLZ_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT612/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h0_DLZ_VT642:=.modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h0_DLZ_VT642_A70:=.modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A70/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h0_DLZ_VT642_A72:=.modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A72/sqrt(.modelle.gesamtnetzwerk.durchlauf);
--MT
h0_MT_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_MT_VT612/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h0_MT_VT642:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_MT_VT642/sqrt(.modelle.gesamtnetzwerk.durchlauf);
--OTD
h0_OTD_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_OTD_VT612/sqrt(.modelle.gesamtnetzwerk.durchlauf);
h0_OTD_VT642:= .modelle.gesamtnetzwerk.t_alfa2_n*.modelle.gesamtnetzwerk.standard_deviation_OTD_VT642/sqrt(.modelle.gesamtnetzwerk.durchlauf);
-- Copy the values and compute the plots after each iteration
.modelle.gesamtnetzwerk.Data_collector;
```

Figure 5-25 Computation of the limits for half-width method

When the initial number of simulation runs is reached, the method computes the number of simulation runs required for each variable and find the maximum as in the t-iterative method Figure 5-26.

```
    -- Once that h0 has been computed, we calculate the number of necesaries replications with the formula n=n0*(h0/h)^2
    if (.modelle.gesamtnetzwerk.durchlauf == .modelle.gesamtnetzwerk.Initial_Durchlauf) then
        HW_n_replications_DLZ_VT612:= .modelle.gesamtnetzwerk.Initial_Durchlauf*pow(h0_DLZ_VT612/userDefined_limit_DLZ,2);
        HW_n_replications_DLZ_VT642:= .modelle.gesamtnetzwerk.Initial_Durchlauf*pow(h0_DLZ_VT642/userDefined_limit_DLZ,2);
        HW_n_replications_DLZ_VT642_A70:=.modelle.gesamtnetzwerk.Initial_Durchlauf*pow(h0_DLZ_VT642_A70/userDefined_limit_DLZ,2);
        HW_n_replications_DLZ_VT642_A72:=.modelle.gesamtnetzwerk.Initial_Durchlauf*pow(h0_DLZ_VT642_A72/userDefined_limit_DLZ,2);
        HW_n_replications_MT_VT612:=.modelle.gesamtnetzwerk.Initial_Durchlauf * pow(h0_MT_VT612/userDefined_limit_MT,2);
        HW_n_replications_MT_VT642:=.modelle.gesamtnetzwerk.Initial_Durchlauf * pow(h0_MT_VT642/userDefined_limit_MT,2);
        HW_n_replications_OTD_VT612:=.modelle.gesamtnetzwerk.Initial_Durchlauf* pow(h0_OTD_VT612/userDefined_limit_OTD,2);
        HW_n_replications_OTD_VT642:=.modelle.gesamtnetzwerk.Initial_Durchlauf* pow(h0_OTD_VT642/userDefined_limit_OTD,2);
```

```
-- Search for the maximum of N
        .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:= HW_n_replications_DLZ_VT612;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<HW_n_replications_DLZ_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=HW_n_replications_DLZ_VT642;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<HW_n_replications_DLZ_VT642_A70 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=HW_n_replications_DLZ_VT642_A70;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<HW_n_replications_DLZ_VT642_A72 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=HW_n_replications_DLZ_VT642_A72;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<HW_n_replications_MT_VT612 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=HW_n_replications_MT_VT612;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<HW_n_replications_MT_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=HW_n_replications_MT_VT642;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<HW_n_replications_OTD_VT612 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=HW_n_replications_OTD_VT612;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<HW_n_replications_OTD_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=HW_n_replications_OTD_VT642;
        end;
    end
end;
```

Figure 5-26 Number of required replications in half-width method

5.2.3 Ratio-Method

The frame for the ratio method is presented in Figure 5-27. The ratio method is based on the premise that the ratio between the standard deviation and the average remains constant. This is a fact that will be checked in the validation on the method when studying why it works or not.

Figure 5-27 Frame of the ratio method

The implementation includes a different way of computing the errors in order to have them in %. As it is based on this ratio assumption, some local variables are declared in the method and shown in Figure 5-28.

```
-- DLZ Ratios
CV_DLZ_VT612:real;
CV_DLZ_VT642:real;
CV_DLZ_VT642_A70:real;
CV_DLZ_VT642_A72:real;
-- OTD Ratios
CV_OTD_VT612:real;
CV_OTD_VT642:real;
-- MT Ratios
CV_MT_VT612:real;
CV_MT_VT642:real;
```

Figure 5-28 Local variables in the Ratio Method

Then comes the computation of the averages as in Figure 5-18 however, while in the t-iterative, the errors are just in the same units, this method requires the errors in % of the average therefore they are computed after calculating the average on each replication. This computation can be shown in Figure 5-29, where the absolutes values of the error given by the user are divided by the average.

```
-- Calculate the value in % of the average

UserDefined_limit_DLZ:=.modelle.gesamtnetzwerk.Max_Error_Durchlaufzeit*86400/.modelle.gesamtnetzwerk.average_DLZ_VT612;
UserDefined_limit_MT:=.modelle.gesamtnetzwerk.Max_Error_Mean_Tardiness*86400/.modelle.gesamtnetzwerk.average_MT_VT612;
UserDefined_limit_OTD:=.modelle.gesamtnetzwerk.Max_Error_OnTime_Delivery/100;
```

Figure 5-29 Errors Transformation in the ratio method

Later on, the standard deviation is calculated exactly as in Figure 5-19 and moreover the CV ratios for each variable and again the "*Data_collector*" is called (Figure 5-30).

```
  -- Calculate the differents CV ratios for the initial set of simulation runs
  CV_DLZ_VT612:=.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT612/.modelle.gesamtnetzwerk.average_DLZ_VT612;
  CV_DLZ_VT642:=.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642/.modelle.gesamtnetzwerk.average_DLZ_VT642;
  CV_DLZ_VT642_A70:=.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A70/.modelle.gesamtnetzwerk.average_DLZ_VT642_A70;
  CV_DLZ_VT642_A72:=.modelle.gesamtnetzwerk.standard_deviation_DLZ_VT642_A72/.modelle.gesamtnetzwerk.average_DLZ_VT642_A72;

  CV_MT_VT612:=.modelle.gesamtnetzwerk.standard_deviation_MT_VT612/.modelle.gesamtnetzwerk.average_MT_VT612;
  CV_MT_VT642:=.modelle.gesamtnetzwerk.standard_deviation_MT_VT642/.modelle.gesamtnetzwerk.average_MT_VT642;

  CV_OTD_VT612:=.modelle.gesamtnetzwerk.standard_deviation_OTD_VT612/.modelle.gesamtnetzwerk.average_OTD_VT612;
  CV_OTD_VT642:=.modelle.gesamtnetzwerk.standard_deviation_OTD_VT642/.modelle.gesamtnetzwerk.average_OTD_VT642;

-- Copy the values and compute the plots after each iteration
  .modelle.gesamtnetzwerk.Data_collector;
```

Figure 5-30 CV ratios in the ratio-method

Finally the strategy is the same as in the other methods, the computation of the number of simulation runs when the initial number is reached and the search for the maximum of all of them to calculate the *"Limit_Durchlauf_Global"* is depicted in Figure 5-31.

```
    if .modelle.gesamtnetzwerk.durchlauf == .modelle.gesamtnetzwerk.Initial_Durchlauf then

        RM_n_replications_DLZ_VT612:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_DLZ*CV_DLZ_VT612,2);
        RM_n_replications_DLZ_VT642:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_DLZ*CV_DLZ_VT642,2);
        RM_n_replications_DLZ_VT642_A70:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_DLZ*CV_DLZ_VT642_A70,2);
        RM_n_replications_DLZ_VT642_A72:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_DLZ*CV_DLZ_VT642_A72,2);
        RM_n_replications_MT_VT612:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_MT*CV_MT_VT612,2);
        RM_n_replications_MT_VT642:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_MT*CV_MT_VT642,2);
        RM_n_replications_OTD_VT612:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_OTD*CV_OTD_VT612,2);
        RM_n_replications_OTD_VT642:= pow(.modelle.gesamtnetzwerk.z_alfa2/userDefined_limit_OTD*CV_OTD_VT642,2);
```

```
-- Search for the maximum of N

        .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:= RM_n_replications_DLZ_VT612;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<RM_n_replications_DLZ_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=RM_n_replications_DLZ_VT642;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<RM_n_replications_DLZ_VT642_A70 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=RM_n_replications_DLZ_VT642_A70;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<RM_n_replications_DLZ_VT642_A72 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=RM_n_replications_DLZ_VT642_A72;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<RM_n_replications_MT_VT612 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=RM_n_replications_MT_VT612;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<RM_n_replications_MT_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=RM_n_replications_MT_VT642;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<RM_n_replications_OTD_VT612 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=RM_n_replications_OTD_VT612;
        end;

        if .modelle.gesamtnetzwerk.Limit_Durchlauf_Global<RM_n_replications_OTD_VT642 then
            .modelle.gesamtnetzwerk.Limit_Durchlauf_Global:=RM_n_replications_OTD_VT642;
        end;
    end;
end;
```

Figure 5-31 Number of Simulation runs in ratio method

5.3 Implementation of Plots and Export of Data

The last part of this section will be the implementation of the outputs. On it, the method programmed to export the data is presented as well as the code necessary to make the plots of the different variables. When the description of the inputs and outputs was made, two outputs were mentioned; values of the simulation runs required to achieve a confidence level and also a plot that shows the value of the average and the limits of the confidence interval with each simulation run. The value of the simulation runs can be easily seen in the frames of each of the simulation methods.

The method starts with the declaration of the local variables; the half-width of the confidence interval for each variable, also the maximum, minimum and the range, in order to transform the variables into % .The code used can be seen in Figure 5-32.

```
-- The obejctive of this method is to copy the values of the averages in a table so we can compute later the plots
is
    i:integer;
        -- DLZ limits
    h_DLZ_VT612:real;
    h_DLZ_VT642:real;
    h_DLZ_VT642_A70:real;
    h_DLZ_VT642_A72:real;

        -- OTD Limits
    h_OTD_VT612:real;
    h_OTD_VT642:real;

        -- MT Limits
    h_MT_VT612:real;
    h_MT_VT642:real;

    -- Minimun Values
    min_DLZ_VT612:real;
    min_DLZ_VT642:real;
    min_DLZ_VT642_A70:real;
    min_DLZ_VT642_A72:real;
    min_OTD_VT612:real;
    min_OTD_VT642:real;
    min_MT_VT612:real;
    min_MT_VT642:real;

    -- Maximum Values
    max_DLZ_VT612:real;
    max_DLZ_VT642:real;
    max_DLZ_VT642_A70:real;
    max_DLZ_VT642_A72:real;
    max_OTD_VT612:real;
    max_OTD_VT642:real;
    max_MT_VT612:real;
    max_MT_VT642:real;

    -- Range Values
    range_DLZ_VT612:real;
    range_DLZ_VT642:real;
    range_DLZ_VT642_A70:real;
    range_DLZ_VT642_A72:real;
    range_OTD_VT612:real;
    range_OTD_VT642:real;
    range_MT_VT612:real;
    range_MT_VT642:real;
```

Figure 5-32 Local variables in the “Data_collector” method

Then since the half-width of the intervals are not a global variable, they need to be computed in the method, besides, the values of the average, the upper limit and the lower limit are stored in a table. This concepts are programmed in the code below Figure 5-33.

```
do

    -- Fix the index to the simulation run number
    i:=durchlauf;

    -- Calculate the limits to create the upper and lower values
    --DLZ
    h_DLZ_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_DLZ_VT612/sqrt(durchlauf);
    h_DLZ_VT642:=.modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_DLZ_VT642/sqrt(durchlauf);
    h_DLZ_VT642_A70:=.modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_DLZ_VT642_A70/sqrt(durchlauf);
    h_DLZ_VT642_A72:=.modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_DLZ_VT642_A72/sqrt(durchlauf);
    --MT
    h_MT_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_MT_VT612/sqrt(durchlauf);
    h_MT_VT642:= .modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_MT_VT642/sqrt(durchlauf);
    --OTD
    h_OTD_VT612:= .modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_OTD_VT612/sqrt(durchlauf);
    h_OTD_VT642:= .modelle.gesamtnetzwerk.t_alfa2_n*standard_deviation_OTD_VT642/sqrt(durchlauf);


        ------- Copy the values in a table DLZ-------
    Plots.Data_Gathering[1,i]:=i;
    Plots.Data_Gathering[2,i]:=Average_DLZ_VT612;
    Plots.Data_Gathering[3,i]:=Average_DLZ_VT612+h_dLZ_VT612;
    Plots.Data_Gathering[4,i]:=Average_DLZ_VT612-h_dlZ_VT612;
    Plots.Data_Gathering[5,i]:=(Average_DLZ_VT642);
    Plots.Data_Gathering[6,i]:=(Average_DLZ_VT642+h_dLZ_VT642);
    Plots.Data_Gathering[7,i]:=(Average_DLZ_VT642-h_dlZ_VT642);
    Plots.Data_Gathering[8,i]:=(Average_DLZ_VT642_A70);
    Plots.Data_Gathering[9,i]:=(Average_DLZ_VT642_A70+h_dLZ_VT642_A70);
    Plots.Data_Gathering[10,i]:=(Average_DLZ_VT642_A70-h_dlZ_VT642_A70);
    Plots.Data_Gathering[11,i]:=(Average_DLZ_VT642_A72);
    Plots.Data_Gathering[12,i]:=(Average_DLZ_VT642_A72+h_DLZ_VT642_A72);
    Plots.Data_Gathering[13,i]:=(Average_DLZ_VT642_A72-h_DLZ_VT642_A72);


        ------- Copy the values in a table MT-------
    Plots.Data_Gathering[14,i]:=Average_MT_VT612;
    Plots.Data_Gathering[15,i]:=Average_MT_VT612+h_MT_VT612;
    Plots.Data_Gathering[16,i]:=Average_MT_VT612-h_MT_VT612;
    if Plots.Data_Gathering[16,i]<0 then -- it does not store negative values
        Plots.Data_Gathering[16,i]:=0;
    end;
        Plots.Data_Gathering[17,i]:=(Average_MT_VT642);
    Plots.Data_Gathering[18,i]:=(Average_MT_VT642+h_MT_VT642);
    Plots.Data_Gathering[19,i]:=(Average_MT_VT642-h_MT_VT642);
    if Plots.Data_Gathering[19,i]<0 then -- it does not store negative values
        Plots.Data_Gathering[19,i]:=0;
    end;

        ------- Copy the values in a table OTD -------
    Plots.Data_Gathering[20,i]:=Average_OTD_VT612;
    Plots.Data_Gathering[21,i]:=Average_OTD_VT612+h_OTD_VT612;
    Plots.Data_Gathering[22,i]:=Average_OTD_VT612-h_OTD_VT612;
    if Plots.Data_Gathering[22,i]<0 then
        Plots.Data_Gathering[22,i]:=0;
    end;

    Plots.Data_Gathering[23,i]:=(Average_OTD_VT642);
    Plots.Data_Gathering[24,i]:=(Average_OTD_VT642+h_OTD_VT642);
    Plots.Data_Gathering[25,i]:=(Average_OTD_VT642-h_OTD_VT642);
    if Plots.Data_Gathering[25,i]<0 then
        Plots.Data_Gathering[25,i]:=0;
    end;
```

Figure 5-33 Storage of real values and confidence interval computation

Now the values are transformed into %, to do so, the maximum and minimum values of the stored data are detected and with them the range of each variable is calculated. The formula used to normalize the results is the following:

$$Variable\ in\ \% = \frac{Variable_{real} - Minimum_{set}}{Range\ of\ the\ data} * 100 \quad (32)$$

The data is transformed only when the validation method is finished. On Figure 5-34 the initialization for the minimum and maximum and the search for them is made.

```
----- Transform into percentages -----
if durchlauf == Limit_Durchlauf_Global then

-- initialice with the first value
min_DLZ_VT612:=Plots.Data_Gathering[4,2];
min_DLZ_VT642:=Plots.Data_Gathering[7,2];
min_DLZ_VT642_A70:=Plots.Data_Gathering[10,2];
min_DLZ_VT642_A72:=Plots.Data_Gathering[13,2];
min_MT_VT612:=plots.data_Gathering[16,2];
min_MT_VT642:=plots.data_Gathering[19,2];
min_OTD_VT612:=plots.data_Gathering[22,2];
min_OTD_VT642:=plots.data_Gathering[25,2];
max_DLZ_VT612:=Plots.Data_Gathering[3,2];
max_DLZ_VT642:=Plots.Data_Gathering[6,2];
max_DLZ_VT642_A70:=Plots.Data_Gathering[9,2];
max_DLZ_VT642_A72:=Plots.Data_Gathering[12,2];
max_MT_VT612:=plots.data_Gathering[15,2];
max_MT_VT642:=plots.data_Gathering[18,2];
max_OTD_VT612:=plots.data_Gathering[21,2];
max_OTD_VT642:=plots.data_Gathering[24,2];

i:=2;
-- search for the minimum and the maximum
-- loop from the initial value in the raw 2 to the last one (durchlauf)
repeat
if min_DLZ_VT612>Plots.Data_Gathering[4,i] then
    min_DLZ_VT612:=Plots.Data_Gathering[4,i];
end;
if min_DLZ_VT642>Plots.Data_Gathering[7,i] then
    min_DLZ_VT642:=Plots.Data_Gathering[7,i]
end;
if min_DLZ_VT642_A70>Plots.Data_Gathering[10,i]then
    min_DLZ_VT642_A70:=Plots.Data_Gathering[10,i];
end;
if min_DLZ_VT642_A72>Plots.Data_Gathering[13,i]then
    min_DLZ_VT642_A72:=Plots.Data_Gathering[13,i];
end;
if min_MT_VT612> plots.data_Gathering[16,i]then
    min_MT_VT612:=plots.data_Gathering[16,i];
end;
if min_MT_VT642> plots.data_Gathering[19,i]then
    min_MT_VT642:=plots.data_Gathering[19,i];
end;
if min_OTD_VT612> plots.data_Gathering[22,i]then
    min_OTD_VT612:=plots.data_Gathering[22,i];
end;
if min_OTD_VT642> plots.data_Gathering[25,i]then
    min_OTD_VT642:=plots.data_Gathering[25,i];
end;
```

```
--- loop to search for the maximum
if max_DLZ_VT612<Plots.Data_Gathering[3,i] then
    max_DLZ_VT612:=Plots.Data_Gathering[3,i];
end;
if max_DLZ_VT642<Plots.Data_Gathering[6,i] then
    max_DLZ_VT642:=Plots.Data_Gathering[6,i]
end;
if max_DLZ_VT642_A70<Plots.Data_Gathering[9,i]then
    max_DLZ_VT642_A70:=Plots.Data_Gathering[9,i];
end;
if max_DLZ_VT642_A72<Plots.Data_Gathering[12,i]then
    max_DLZ_VT642_A72:=Plots.Data_Gathering[12,i];
end;
if max_MT_VT612< plots.data_Gathering[15,i]then
    max_MT_VT612:=plots.data_Gathering[15,i];
end;
if max_MT_VT642< plots.data_Gathering[18,i]then
    max_MT_VT642:=plots.data_Gathering[18,i];
end;
if max_OTD_VT612< plots.data_Gathering[21,i]then
    max_OTD_VT612:=plots.data_Gathering[21,i];
end;
if max_OTD_VT642< plots.data_Gathering[24,i]then
    max_OTD_VT642:=plots.data_Gathering[24,i];
end;

i:=i+1;
until i>=durchlauf;
```

Figure 5-34 Initialization and computation of max and min in the data collector

Once that the max and min of each set of data have been computed, the method calculates the range of values by subtracting the maximum and minimum value of the data series. The code can be seen in Figure 5-35.

```
range_DLZ_VT612:= max_DLZ_VT612-min_DLZ_VT612;
range_DLZ_VT642:= max_DLZ_VT642-min_DLZ_VT642;
range_DLZ_VT642_A70:=max_DLZ_VT642_A70-min_DLZ_VT642_A70;
range_DLZ_VT642_A72:=max_DLZ_VT642_A72-min_DLZ_VT642_A72;
range_MT_VT612:=max_MT_VT612-min_MT_VT612;
range_MT_VT642:=max_MT_VT642-min_MT_VT642;
range_OTD_VT612:=max_OTD_VT612-min_OTD_VT612;
range_OTD_VT642:=max_OTD_VT642-min_OTD_VT642;
```

Figure 5-35 Computation of the range of values

Finally the values are computed by using the previous equation and stored in a table called “Data_gathering_percent” and also the plots from the real values are rescaled so the information is shown in the maximal range possible. The code is presented in Figure 5-36

```
    i:=2;
    repeat
            ------- Copy the values in a table DLZ-------
    Plots.Data_Gathering_Percent[1,i]:=i;
    Plots.Data_Gathering_Percent[2,i]:=(Plots.Data_Gathering[2,i]-min_DLZ_VT612)/range_DLZ_VT612*100;
    Plots.Data_Gathering_Percent[3,i]:=(Plots.Data_Gathering[3,i]-min_DLZ_VT612)/range_DLZ_VT612*100;
    Plots.Data_Gathering_Percent[4,i]:=(Plots.Data_Gathering[4,i]-min_DLZ_VT612)/range_DLZ_VT612*100;
    Plots.Data_Gathering_Percent[5,i]:=(Plots.Data_Gathering[5,i]-min_DLZ_VT642)/range_DLZ_VT642*100;
    Plots.Data_Gathering_Percent[6,i]:=(Plots.Data_Gathering[6,i]-min_DLZ_VT642)/range_DLZ_VT642*100;
    Plots.Data_Gathering_Percent[7,i]:=(Plots.Data_Gathering[7,i]-min_DLZ_VT642)/range_DLZ_VT642*100;
    Plots.Data_Gathering_Percent[8,i]:=(Plots.Data_Gathering[8,i]-min_DLZ_VT642_A70)/range_DLZ_VT642_A70*100;
    Plots.Data_Gathering_Percent[9,i]:=(Plots.Data_Gathering[9,i]-min_DLZ_VT642_A70)/range_DLZ_VT642_A70*100;
    Plots.Data_Gathering_Percent[10,i]:=(Plots.Data_Gathering[10,i]-min_DLZ_VT642_A70)/range_DLZ_VT642_A70*100;
    Plots.Data_Gathering_Percent[11,i]:=(Plots.Data_Gathering[11,i]-min_DLZ_VT642_A72)/range_DLZ_VT642_A72*100;
    Plots.Data_Gathering_Percent[12,i]:=(Plots.Data_Gathering[12,i]-min_DLZ_VT642_A72)/range_DLZ_VT642_A72*100;
    Plots.Data_Gathering_Percent[13,i]:=(Plots.Data_Gathering[13,i]-min_DLZ_VT642_A72)/range_DLZ_VT642_A72*100;
        ------- Copy the values in a table MT-------
    Plots.Data_Gathering_Percent[14,i]:=(Plots.Data_Gathering[14,i]-min_MT_VT612)/range_MT_VT612*100;
    Plots.Data_Gathering_Percent[15,i]:=(Plots.Data_Gathering[15,i]-min_MT_VT612)/range_MT_VT612*100;
    Plots.Data_Gathering_Percent[16,i]:=(Plots.Data_Gathering[16,i]-min_MT_VT612)/range_MT_VT612*100;
    if Plots.Data_Gathering_Percent[16,i]<0 then
        Plots.Data_Gathering_Percent[16,i]:=0;
    end;
    Plots.Data_Gathering_Percent[17,i]:=(Plots.Data_Gathering[17,i]-min_MT_VT642)/range_MT_VT642*100;
    Plots.Data_Gathering_Percent[18,i]:=(Plots.Data_Gathering[18,i]-min_MT_VT642)/range_MT_VT642*100;
    Plots.Data_Gathering_Percent[19,i]:=(Plots.Data_Gathering[19,i]-min_MT_VT642)/range_MT_VT642*100;
    if Plots.Data_Gathering_Percent[19,i]<0 then
        Plots.Data_Gathering_Percent[19,i]:=0;
    end;
        ------- Copy the values in a table OTD -------
    Plots.Data_Gathering_Percent[20,i]:=(Plots.Data_Gathering[20,i]-min_OTD_VT612)/range_OTD_VT612*100;
    Plots.Data_Gathering_Percent[21,i]:=(Plots.Data_Gathering[21,i]-min_OTD_VT612)/range_OTD_VT612*100;
    Plots.Data_Gathering_Percent[22,i]:=(Plots.Data_Gathering[22,i]-min_OTD_VT612)/range_OTD_VT612*100;
    if Plots.Data_Gathering_Percent[22,i]<0 then
        Plots.Data_Gathering_Percent[22,i]:=0;
    end;
    Plots.Data_Gathering_Percent[23,i]:=(Plots.Data_Gathering[23,i]-min_OTD_VT642)/range_OTD_VT642*100;
    Plots.Data_Gathering_Percent[24,i]:=(Plots.Data_Gathering[24,i]-min_OTD_VT642)/range_OTD_VT642*100;
    Plots.Data_Gathering_Percent[25,i]:=(Plots.Data_Gathering[25,i]-min_OTD_VT642)/range_OTD_VT642*100;
    if Plots.Data_Gathering_Percent[25,i]<0 then
        Plots.Data_Gathering_Percent[25,i]:=0;
    end;
    i:=i+1;
    until i>=durchlauf;


        -- Reescale the plots
    Plots.DLZ_VT612_real.ySkalaMin:=min_DLZ_VT612;
    Plots.DLZ_VT642_real.ySkalaMin:=min_DLZ_VT642;
    Plots.DLZ_VT642_A70_real.ySkalaMin:=min_DLZ_VT642_A70;
    Plots.DLZ_VT642_A72_real.ySkalaMin:=min_DLZ_VT642_A72;
    Plots.MT_VT612_real.ySkalaMin:= min_MT_VT612;
    Plots.MT_VT642_real.ySkalaMin:= min_MT_VT642;
    Plots.OTD_VT612_real.ySkalaMin:= min_OTD_VT612;
    Plots.OTD_VT642_real.ySkalaMin:= min_OTD_VT642;
    end;
end;
```

Figure 5-36 Store the scaled values and rescale of the real plots

The other method present in the output is the “export_to_excel” method. When run, the method create two excel files, one with the data from the average values of the simulation run and another with the values of the plots. Hereunder the code can be seen in Figure 5-37.

```
is
do
    DLZ_Mittelwerte.schreibeExcelDatei("Z:\ModellCopy\Validation.xlsx","DLZ_mittlewert");
    Mittelwert_Verspätung_612.schreibeExcelDatei("Z:\ModellCopy\Validation.xlsx","Mittelwert_Verspätung_612");
    Mittelwert_Verspätung_642.schreibeExcelDatei("Z:\ModellCopy\Validation.xlsx","Mittelwert_Verspätung_642");
    Plots.schreibeExcelDatei("Z:\ModellCopy\Plots.xlsx");
end;
```

Figure 5-37 Export to Excel method

6 VALIDATION AND RESULTS OF THE METHODS

The section describes the methods implemented in an excel file. Since we can export the data, we could check if the methods are working as they should or not. Furthermore the section describes an experiment to evaluate the performance of the methods. In this experiment, the validation model runs the confidence levels of 90%-95%-97%, to do so, the validation model was edited to evaluate everything with the same data set. Later on, another experiment has been made by varying the errors assumed and checking again the variation in the different confidence levels.

6.1 Implementation in Excel File

By running the method "*Export_to_Excel*" the model provides us with an excel file that contains the real data of three tables "*DLZ_Mittelwert*" or average cycle time, "Mittelwert_verspätung_612" and "Mittelwert_verspätung_642" that contains the mean tardiness "*Mittlere_verspätung*" and the on-time delivery "*Termintreue*". Once that the data is collected, the different, methods have been computed both analytically and using the statistical tools of excel to make a double check of the proper working.

- **Validation of T-Iterative Method**

To check the validation of the T-Iterative we have run the model and then we have check different values of the sample size. The values given in the output provides the minimum of simulation runs necessaries to validate the method. When the model was run, it stopped at 96 simulation runs therefore that will be the first value to check.

		Significance Level		Sample Size		
2		0,05		96		
3	Variable	Mean	Std. Deviation		Confidence Interval	IS IT VALID?
4	DLZ_VT612	1004100,669	28693,1		5813,762	✓ 11466,24
5	DLZ_VT642	971665,01	46491,02		9419,957	✓ 7860,043
6	DLZ_VT642_A70	783160,4652	45668,84		9253,367	✓ 8026,633
7	DLZ_VT642_A72	1106813,927	52870,55		10712,57	✓ 6567,431
8	MT_VT612	1,83498249	1,4443		0,292642	✓ 0,007358
9	MT_VT642	2,327604637	0,64683		0,13106	✓ 0,16894
10	OTD_VT612	0,977816358	0,017631		0,003572	✓ 0,011428
11	OTD_VT642	0,831374644	0,057895		0,011731	✓ 0,003269
12						
13						

Figure 6-1 T-Iterative Method results for 96 simulation runs

On Figure 6-1 we can see the validation of the method, the average and std. deviation have been computed by using excel formulas. The “Is it valid?” space, check the confidence interval with the limits sets by the user. If the values is positive, it means that the range of the confidence interval is below our limit thus successful. On Figure 6-2 , we can see the results of the simulation output with the limit fixed by the variable “MT_VT612” at 96 simulation runs.

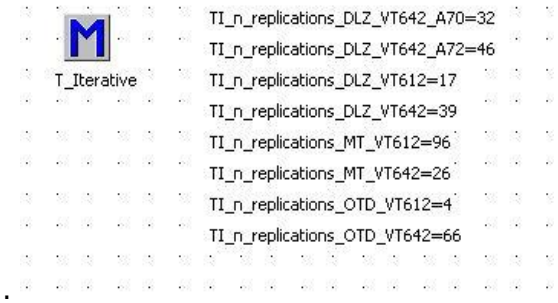


Figure 6-2 Output of Validation

The validation have been checked for another values of simulation runs (40,50,67) all of them with good results; at 40 sim runs, all the variables are validated but “MT_VT612” that needs 96 sim runs “DLZ_VT642_A72” that needs 46 and “OTD_VT642” that needs 66. Notice that the different between one variable validated or not is 30 simulation runs that is around a 30% of the time.

	Significance Level	Sample Size		
	0,05	40		
Variable	**Mean**	**Std. Deviation**	**Confidence Interval**	**IS IT VALID?**
DLZ_VT612	2384051,02	29635,94	9478,034	✓ 7801,966
DLZ_VT642	2307692,717	53599,41	17141,92	✓ 138,0757
DLZ_VT642_A70	1858780,079	45951,17	14695,9	✓ 2584,103
DLZ_VT642_A72	2629701,28	61961,7	19816,31	✗ -2536,31
MT_VT612	4,287651853	1,157237	0,370102	✗ -0,0701
MT_VT642	5,529255825	0,670788	0,214528	✓ 0,085472
OTD_VT612	2,321990741	0,017058	0,005456	✓ 0,009544
OTD_VT642	1,973717949	0,066874	0,021387	✗ -0,006387

		Significance Level	Sample Size		
		0,05	50		
17	**Variable**	**Mean**	**Std. Deviation**	**Confidence Interval**	**IS IT VALID?**
18	DLZ_VT612	1907240,816	30178,74	8576,703	✓ 8703,297
19	DLZ_VT642	1846154,173	50402,47	14324,22	✓ 2955,775
20	DLZ_VT642_A70	1487024,063	47533,72	13508,93	✓ 3771,066
21	DLZ_VT642_A72	2103761,024	57805,5	16428,14	✓ 851,8585
22	MT_VT612	3,430121482	1,167615	0,331832	✗ -0,03183
23	MT_VT642	4,42340466	0,646373	0,183697	✓ 0,116303
24	OTD_VT612	1,857592593	0,018539	0,005269	✓ 0,009731
25	OTD_VT642	1,578974359	0,063546	0,01806	✗ -0,00306

	Significance Level		Sample Size		
	0,05		67		
Variable	Mean	Std. Deviation		Confidence Interval	IS IT VALID?
DLZ_VT612	1423314,042	28849,41		7036,924	✓ 10243,08
DLZ_VT642	1377726,995	49753,27		12135,78	✓ 5144,223
DLZ_VT642_A70	1109719,45	48952,24		11940,39	✓ 5339,61
DLZ_VT642_A72	1569970,913	56196,41		13707,38	✓ 3572,62
MT_VT612	2,559792151	1,535384		0,374509	✗ -0,07451
MT_VT642	3,301048254	0,64836		0,158147	✓ 0,141853
OTD_VT612	1,386263129	0,018499		0,004512	✓ 0,010488
OTD_VT642	1,178339074	0,061091		0,014901	✓ 0,000099

Figure 6-3 Output of simulation runs for different sample sizes

In addition to the validation of the method and since it is the method that represents the iteration until the result is achieved the work presents now the plots for the different confidence intervals. Please notice that the plots are in % so it may lead to confusion since the range depends on the maximum and minimum of the data. The % has been computed in such a way that maximizes the visualization of the curve, to if the confidence is achieved or not. Hereunder are represented the plots for the 8 variables measured in the Figure 6-4 to Figure 6-10.

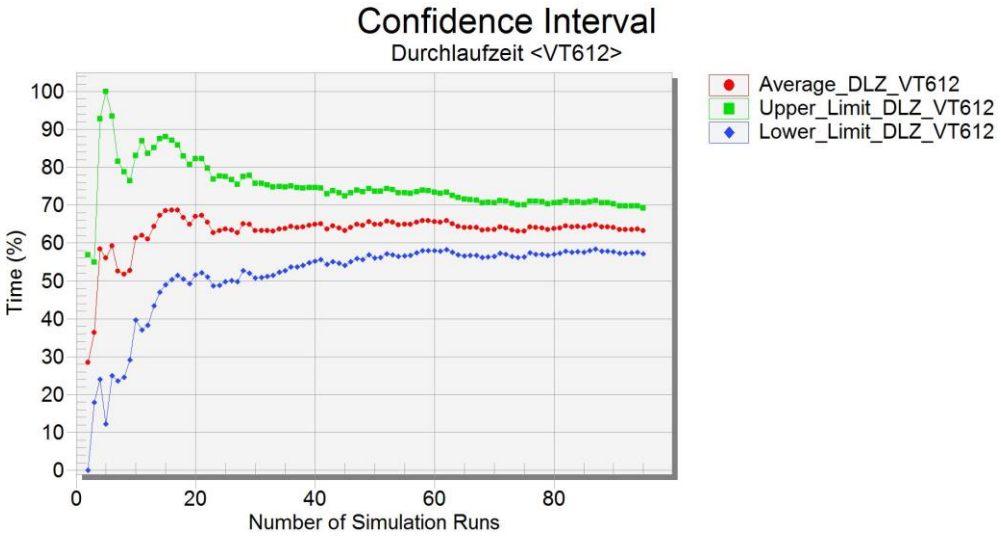


Figure 6-4 Output DLZ_VT612

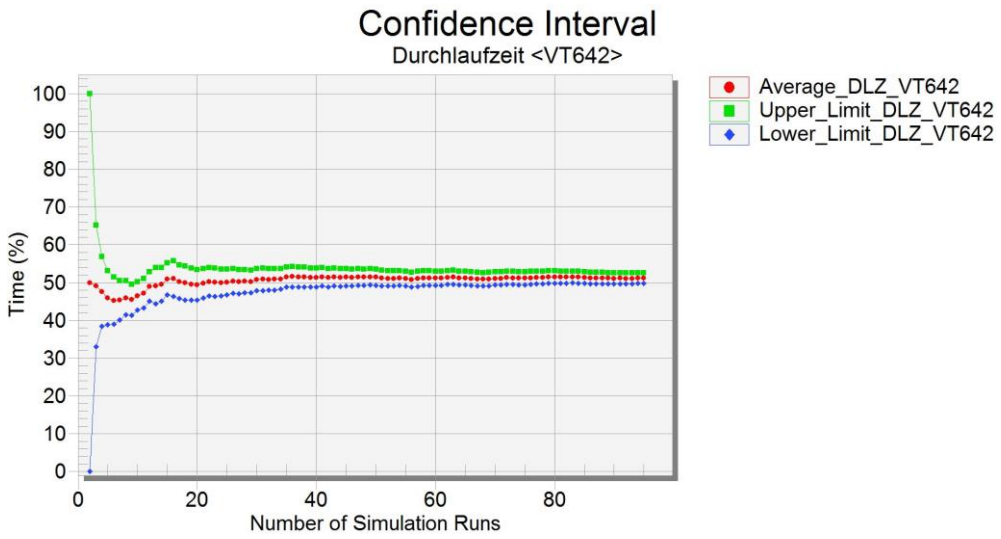


Figure 6-5 Output DLZ_VT642

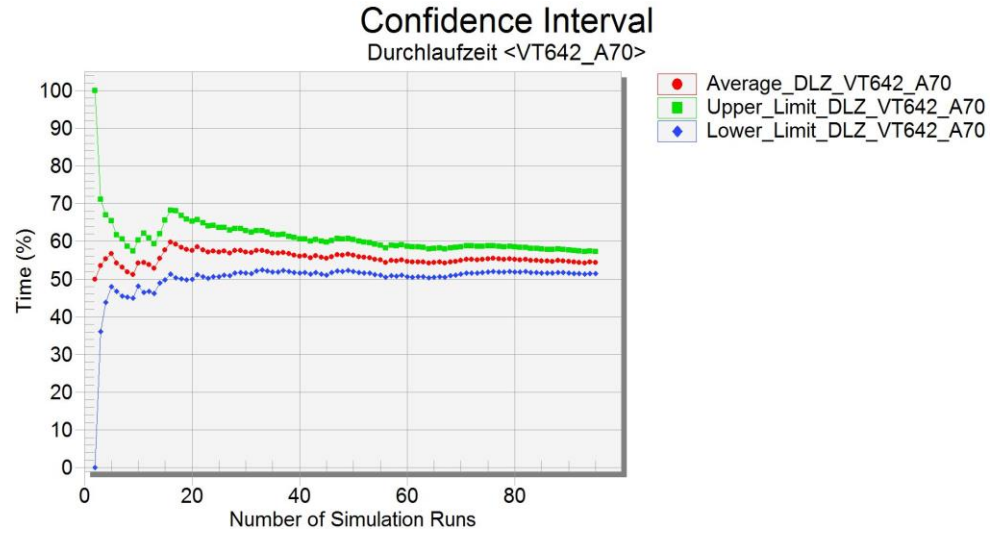


Figure 6-6 Output DLZ_VT642_A70

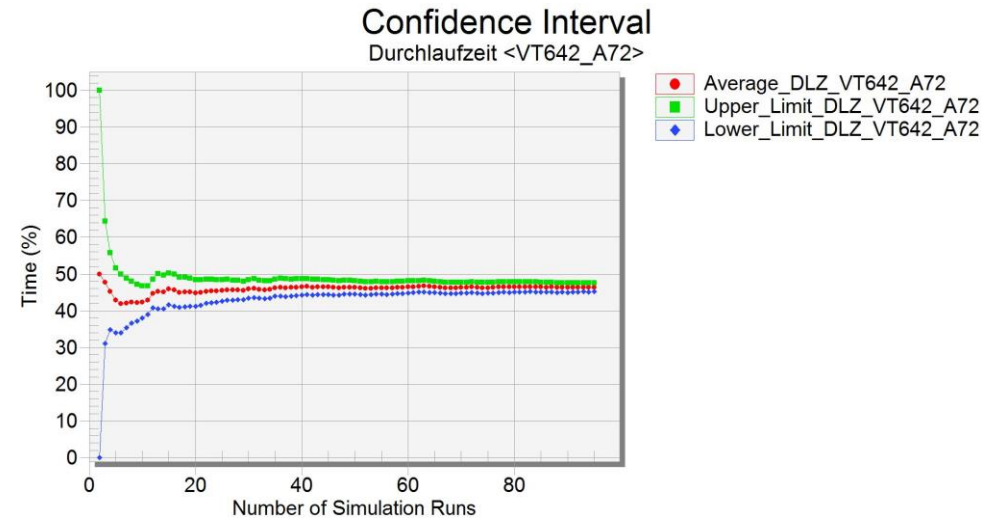


Figure 6-7 Output DLZ_VT642_A72

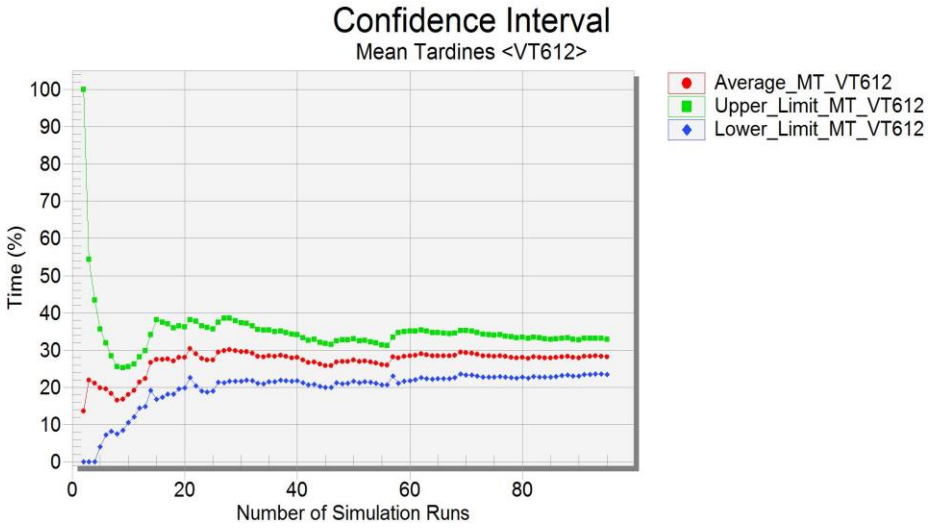


Figure 6-8 Output MT_VT612

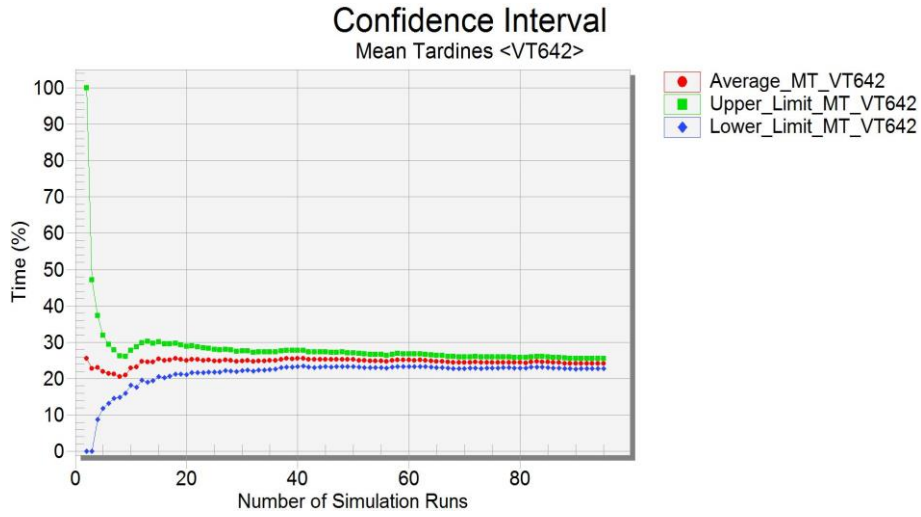


Figure 6-9 Output MT_VT642

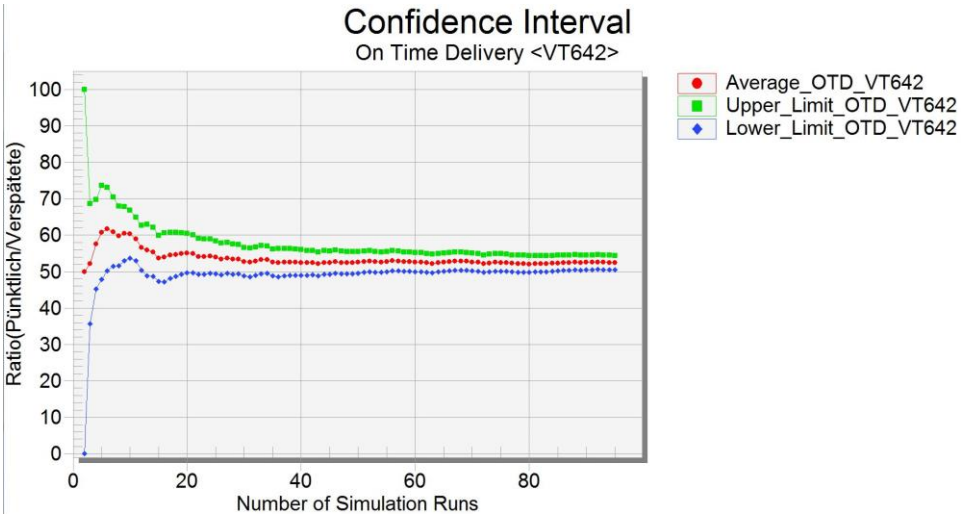


Figure 6-10 Output OTD_VT642

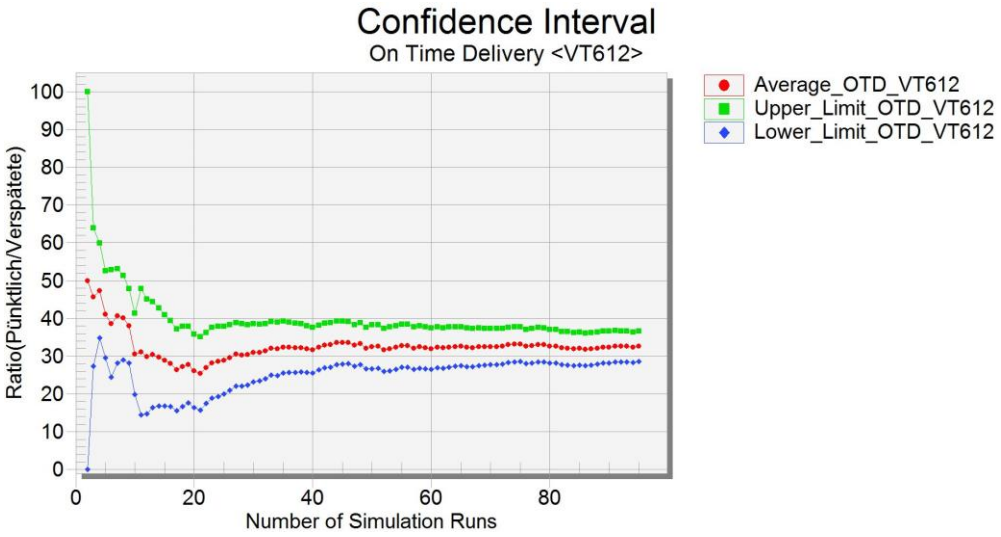


Figure 6-11 Output OTD_VT612

- **Validation of Half-Width method**

For the validation of the half-width method, we have computed the value of h_0 in excel and then we have calculated the approximated number of simulation runs with the formula presented in the description of the method. In the Figure 6-12 both results have been depicted and as it can be seen the results are the same.

	Significance Level	Initial n of sim runs	
	0,05	20	
Variable	**Std. Deviation**	**h0**	**Sim. Run adviced**
DLZ_VT612	33018,19632	15452,9916	14
DLZ_VT642	60138,05393	28145,4756	50
DLZ_VT642_A70	53448,74206	25014,7813	39
DLZ_VT642_A72	69707,46576	32624,0982	66
MT_VT612	1,063036568	0,49751643	60
MT_VT642	0,797876908	0,37341789	30
OTD_VT612	0,019783489	0,00925896	6
OTD_VT642	0,070995431	0,03322688	92
Limits		**Limits in Seconds**	
DLZ (Days)	0,2	17280	
OTD (%)	1%		
MT (Days)	0,3	25920	

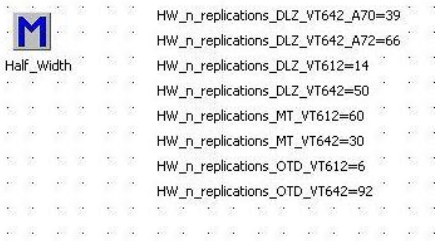


Figure 6-12 Validation of the half-width method

The interesting part comes when comparing this method with the t-iterative. The method half-width works and gives us an idea of the number of simulation runs that needs to be computed. This approximation is better when the average is more or less a stable value. The expected result for this method was not to get an exact value but a value that is over the number of simulation runs for the t-iterative method. Below, in the Figure 6-13, we can see the comparative between the two models.

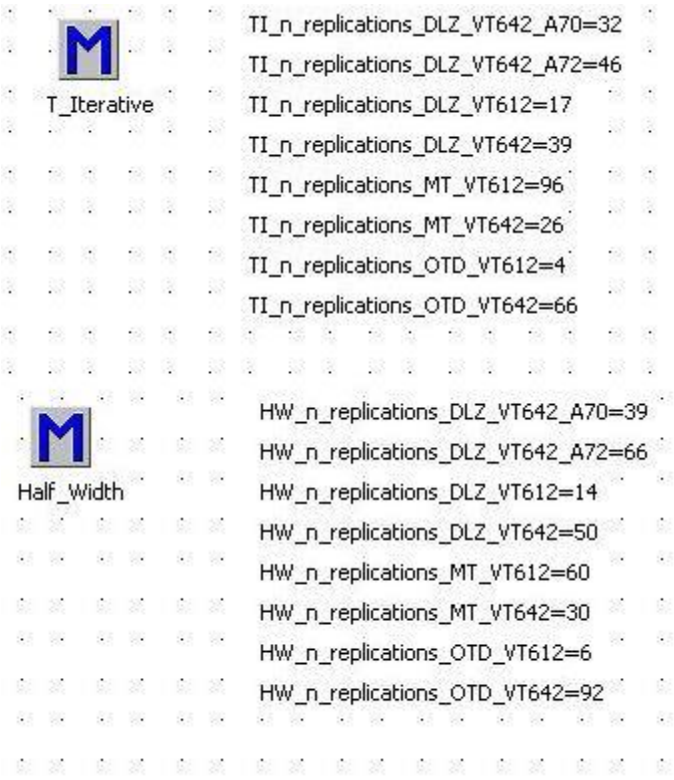


Figure 6-13 Comparative between the model t-iterative and the half-width

The model fails when computing the value for the variable <MT_612> but it also true that this value works in a small range of values and that the average is not stable at 20 simulation runs and leads to long number of simulation runs. (Figure 6-8).

- **Validation of Ratio-Method**

The validation for the ratio method follows the same style as the half-width. However, the first thing that we need to check, because it was an assumption made in the method is if the CV remains constant during the process. In the Figure 6-14 we can see the validation of the method.

	Significance Level		Sample Size						
	0,05		96					INITIAL Nº 20	
Variable	Mean	Std. Deviation		Confidence Interval		Limit	Stand deviati	CV_initial_sim_run	Sim. Run adviced
DLZ_VT612	1004100,669	0,028576		0,00579		1,72%	0,004066993	0,031961338	9
DLZ_VT642	971665,01	46491,02		9419,957		1,78%	0,005372279	0,062198848	33
DLZ_VT642_A70	783160,4652	45668,84		9253,367		2,21%	0,007448315	0,067295167	25
DLZ_VT642_A72	1106813,927	52870,55		10712,57		1,56%	0,007259168	0,062288512	43
MT_VT612	1,83498249	1,4443		0,292642		16,35%	0,099246888	0,618692483	39
MT_VT642	2,327604637	0,64683		0,13106		12,89%	0,060445696	0,33142301	18
OTD_VT612	0,977816358	0,017631		0,003572		1,02%	0,002231713	0,020087208	10
OTD_VT642	0,831374644	0,057895		0,011731		1,20%	0,005873719	0,082134086	126
				z_value			Limits		%
				1,644854			DLZ (Days)	0,20	2984446,369
							OTD (%)	1%	1%
							MT (Days)	0,30	0,163489298

Figure 6-14 Validation of the ratio method

To evaluate the variability of the CV it has been computed in the excel file and then the standard deviation has been computed. We have calculated the standard deviation since it is a measure of the variability of a variable. As seen in the previous figure, the higher value of standard deviation is hold by the variable <MT_VT612> and this is a result that do not surprise us since the results given by the previous methods (the higher the variability, the higher the number of replications necessaries to achieve the confidence desired). As a conclusion for this method, we could say that the method does not work since almost all the values of the method are below the values that should be achieved with the t-iterative. Moreover, in the literature [Byr13] , the author does not consider a limit where the ratio is defined as constant or not. But what it is obvious it that it does not fulfil the expectations.

6.2 Evaluation of the method's performance

With the aim of rating the performance of the methods, a modification of the validation model was made. Now, the model runs until the confidence of 97% is achieved but in the way to do it, it records the values for different confidence levels (90%-95%). The modification is made so we work all the time with the same data because running the previous model just with different levels of confidence will also provide different input data and the results will not be useful. The main changes made where to triplicate some variables such as the confidence levels, or the values of the tables (Figure 6-15)

t_alfa2_n=3.18244630528371	z_alfa2=1.95996398454005
t_alfa2_n_95=3.18244630528371	z_alfa2_90=1.95996398454005
t_alfa2_n_90=3.18244630528371	z_alfa2_95=1.95996398454005

Figure 6-15 Modification on the validation method, input variables

Regarding the t-iterative method, we had to triplicate the output variables as seen in Figure 6-16 and the limits computed in the method.

TI_n_replications_DLZ_VT612=2	TI_n_replications_DLZ_VT612_CL90=2	TI_n_replications_DLZ_VT612_CL95=2
TI_n_replications_DLZ_VT642=2	TI_n_replications_DLZ_VT642_CL90=2	TI_n_replications_DLZ_VT642_CL95=2
TI_n_replications_DLZ_VT642_A70=2	TI_n_replications_DLZ_VT642_A70_CL90=2	TI_n_replications_DLZ_VT642_A70_CL95=2
TI_n_replications_DLZ_VT642_A72=2	TI_n_replications_DLZ_VT642_A72_CL90=2	TI_n_replications_DLZ_VT642_A72_CL95=2
TI_n_replications_MT_VT612=2	TI_n_replications_MT_VT612_CL90=2	TI_n_replications_MT_VT612_CL95=2
TI_n_replications_MT_VT642=2	TI_n_replications_MT_VT642_CL90=2	TI_n_replications_MT_VT642_CL95=2
TI_n_replications_OTD_VT612=2	TI_n_replications_OTD_VT612_CL90=2	TI_n_replications_OTD_VT612_CL95=2
TI_n_replications_OTD_VT642=2	TI_n_replications_OTD_VT642_CL90=2	TI_n_replications_OTD_VT642_CL95=2

Figure 6-16 Output variables of the t-iterative method, modified version

```
------LIMITS FOR 97%------
-- DLZ limits
h_DLZ_VT612:real;
h_DLZ_VT642:real;
h_DLZ_VT642_A70:real;
h_DLZ_VT642_A72:real;
-- OTD Limits
h_OTD_VT612:real;
h_OTD_VT642:real;
-- MT Limits
h_MT_VT612:real;
h_MT_VT642:real;

------LIMITS FOR 90%------
-- DLZ limits
h_DLZ_VT612_CL90:real;
h_DLZ_VT642_CL90:real;
h_DLZ_VT642_A70_CL90:real;
h_DLZ_VT642_A72_CL90:real;
-- OTD Limits
h_OTD_VT612_CL90:real;
h_OTD_VT642_CL90:real;
-- MT Limits
h_MT_VT612_CL90:real;
h_MT_VT642_CL90:real;

------LIMITS FOR 95%------
-- DLZ limits
h_DLZ_VT612_CL95:real;
h_DLZ_VT642_CL95:real;
h_DLZ_VT642_A70_CL95:real;
h_DLZ_VT642_A72_CL95:real;
-- OTD Limits
h_OTD_VT612_CL95:real;
h_OTD_VT642_CL95:real;
-- MT Limits
h_MT_VT612_CL95:real;
h_MT_VT642_CL95:real;
```

Figure 6-17 Limits in the modified version of the t-iterative method

Regarding the half width, we also had to triplicate the output variables as seen in Figure 6-18, so we will have 3 results for each variable (90%, 95%, and 97%).

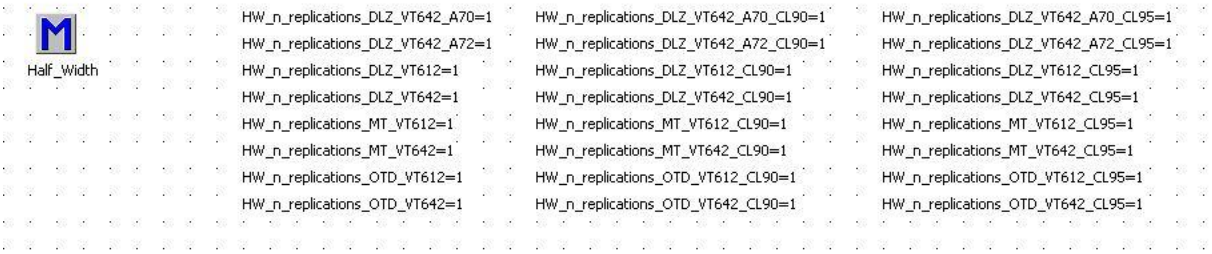


Figure 6-18 Output variables of the half-width method, modified version

Finally some other aspects of the code were modified such as the increment of the variables when the limit is not reached in the t-iterative method and the computations of the half-width once that the initial number of simulation runs is reached.

Furthermore, from the previous results of the validation, we have notice that the value that could create a significant difference in the number of simulation runs is the limits fixed in the Mean tardiness, thus, the modified validation model has been run with different values of this limit in order to show how a small differences here could lead in shorter simulation times.

First of all the model is run with the following limits:

Variable	Value	Unit
Durchlaufzeit	0.15	Days
On-Time Delivery	1.5	%
Mean Tardiness*	0.2	Days

Table 6-1 Input Values of the first experiment

The results for each variable is shown in the next figure (Figure 6-19). The graph below, shows the number of necessary simulation runs for a value of 0.2 days in the mean tardiness. As it can be seen, the variable <MT_VT642> marks the number of simulation runs and while the others are quite quick this variable require a higher number of simulation runs. A lecture from this graph could be that its worthy to run only 101 times since all the variables are validated at 97% and only one at 90%.The changes on this variable have been thought in a sense of reducing the number of simulation runs without losing precision.

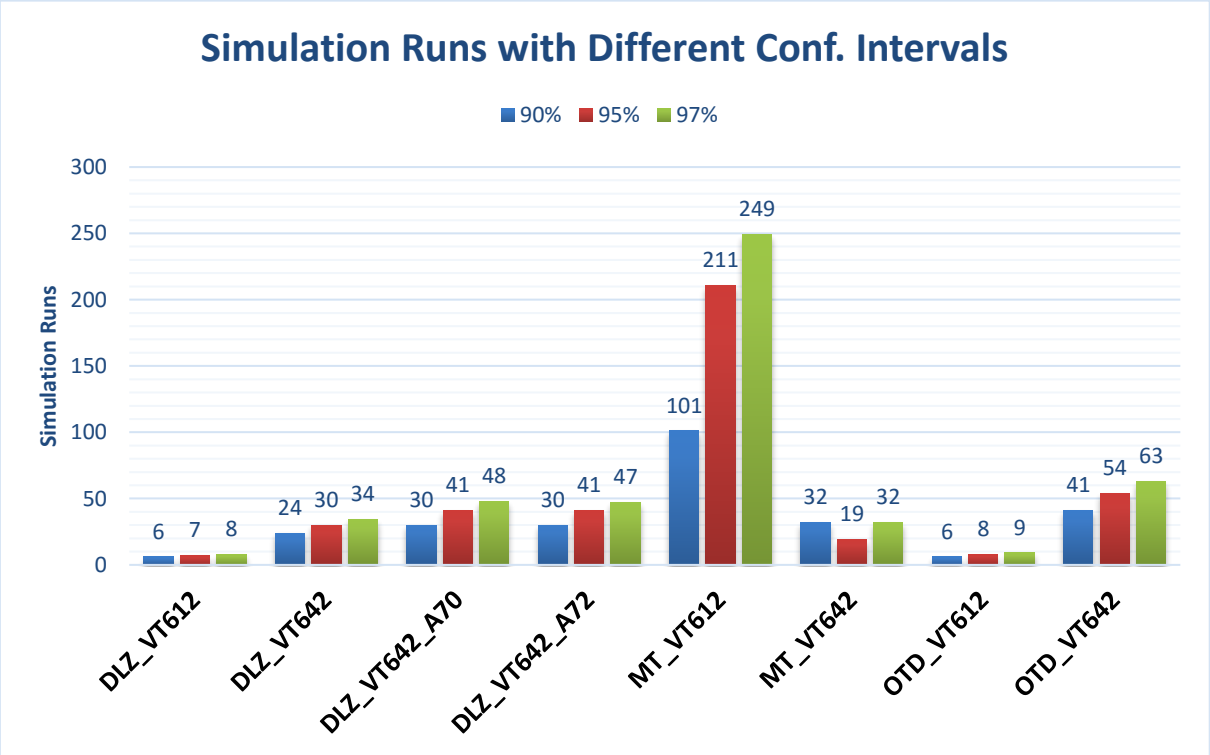


Figure 6-19 Outputs with MT limit=0.2 Days

The second experiment inputs are in the table below:

Variable	Value	Unit
Durchlaufzeit	0.15	Days
On-Time Delivery	1.5	%
Mean Tardiness*	0.25	Days

Table 6-2 Input values of the second experiment

The results from the second experiment are depicted below (Figure 6-20). As was expected, the number of simulation runs of the variables MT_<traincode> have been reduced. It's important to say that the grater the limits, the higher the error that we are assuming. We could have decided to aim the experiment into the other direction and reduce the limit of the other variables to see how precise could we be with the number of simulation runs performed.

To the question of how is it possible to have different limits of simulation runs in variables that we haven't touched, the answer is that every time that we change a parameter, the model is reset. So the values of DLZ, MT and OTD of each simulation run are different but under my experience with the model, comparable.

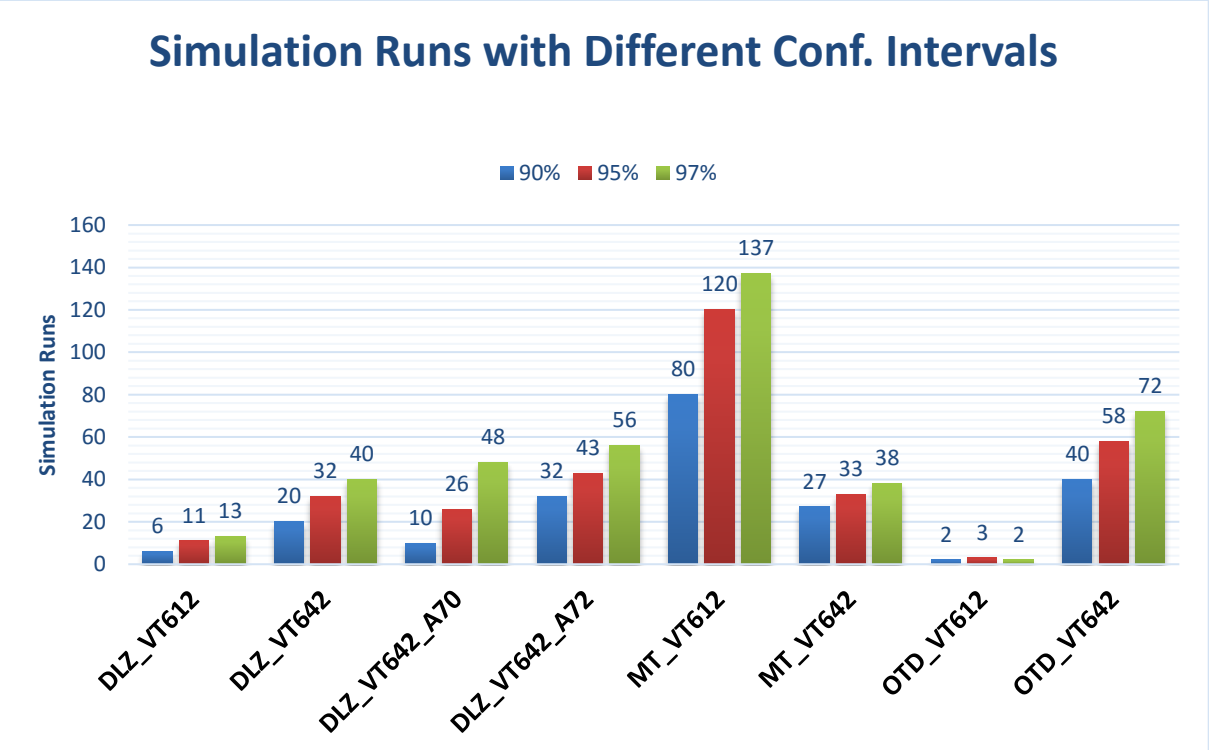


Figure 6-20 Outputs with MT limit=0.25 Days

Finally the third experiment increases even more the limit of the mean tardiness, the inputs are shown in Table 6-3.

Variable	Value	Unit
Durchlaufzeit	0.15	Days
On-Time Delivery	1.5	%
Mean Tardiness*	0.3	Days

Table 6-3 Input Values of the third experiment

The result of the third experiment are expected to be close to the values of the other variables but paying the price of the loss of precision in the measure.

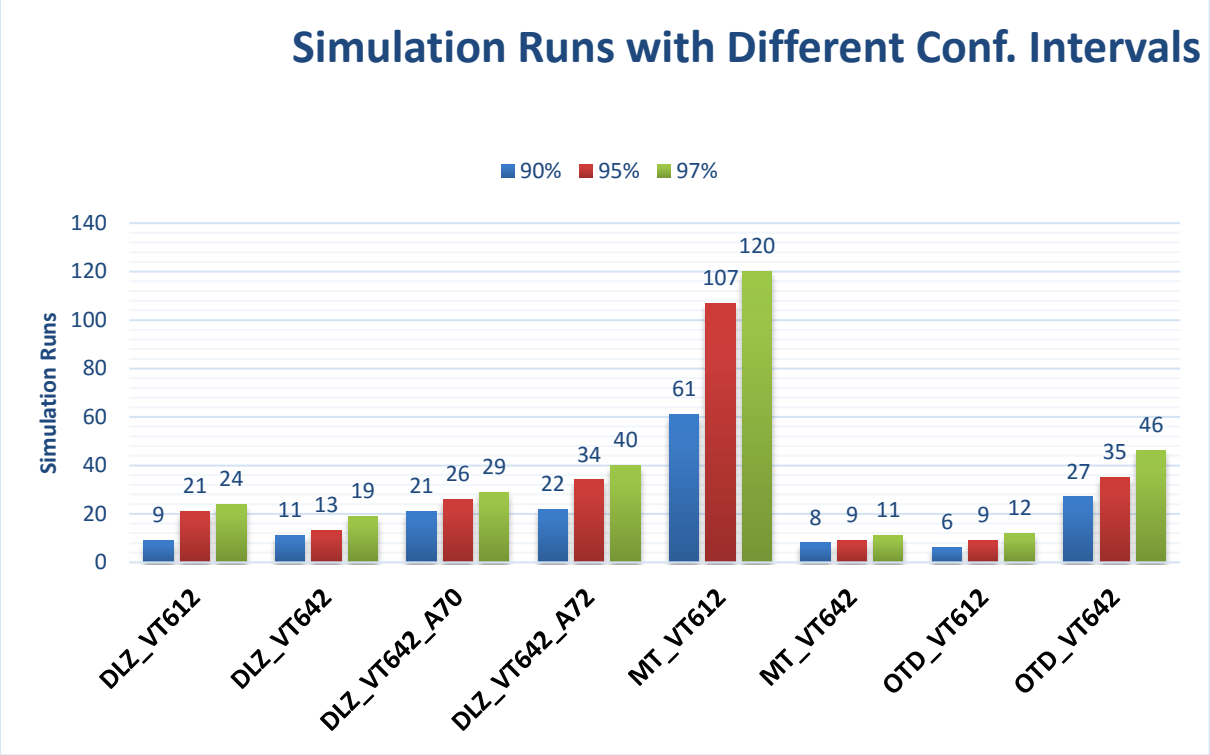


Figure 6-21 Outputs with MT limit=0.3 Days

7 CONCLUSION

This dissertation contains different methods that could be used to calculate and approximate or in some cases an exact number of simulation runs, however, since every simulation is different, we could never assume those values as absolutes. From the 3 methods implemented in the simulation model, one provides an exact result and another gives an early idea of the time that will be needed to validate the model. As they share a lot of variables, they're computation together is a real possibility so the idea will be to run the model with the exact method, wait until the initial number of variables is achieved and at that moment check the half-width method, that will give the user an idea of the number of replications needed and then is up to him/her to decide whether continue with it or maybe is worthy to change some input parameters in order to improve the results or save time.

From the method that does not work, it was tried to perform an analysis and diagnose of the reason for the lack of accuracy, the computation of the standard deviation that could give some light on the topic was completely useless since it does not provide any evidence to be the reason for the inaccuracy.

Finally and related with the experiments carried out, it is clear that one variable is dragging the number of simulation runs to a higher number, however it is also true that this model is not extremely complex and that the time to perform around 250 simulation runs was not longer than 10 minutes. Therefore, the precision and the confidence levels should be maintained. Whether it is worthy or not to change the parameters is always a decision that should be made by the user.

8 FURTHER DEVELOPMENTS

While the development of the methods and without knowing that there was already a variable that control the number of simulation runs. The first idea that came to my mind was to use the experiment manager to control this variable. The value could be accessed from one of the attributes of the object. The problem presented and that should be faced is that when we use the experiment manager, it creates a closed loop that set the number of simulation runs and only present the results after running all of them. It seems not possible, at least from the beginning, to change the number of simulation runs dynamically. However, if this problem is solved, all the statistical tools to analyse and perform confidence intervals become available. Another problem that could be found is that the software is not very flexible when creating this confidence interval. Despite those problems it could be worthy to check this possibility since the software is able to perform all the statistical computations without the necessity of creating new variables (local or global) and programming, the user only have to set the output variable, and the software

9 BIBLIOGRAPHY

[Alp13] Alptekinoglu A.; Banerjee A.; Paul A. et al. (2013): Inventory Pooling to Deliver Differenciated Service. *Manufacturing & Service Operations Management*, 15, 1, 33-44.

[Bake09] Baker K.R. und Trietsch D. (2009): *Principles of Sequencing and Scheduling*.

[Bang10] Bangsow S. (2010): *Manufacturing Simulation with Plant Simulation and SimTalk*. Springer.

[Beck11] Beck S.; Schmidt M. und Nyhuis P. (2011): Controlling the Time Synchronicity of Convergent Supply Processes. *Proceedings of the World Congress on Engineering and Computer Science*, II, 1150-1154.

[Behf15] Behfard S.; van der Heijden M.C.; Al Hanbali A. et al. (2015): Last time buy and repair decisions for spare parts. *European Journal of Operational Research*, 244, 498-510.

[Bier16] Bierer A.; Götze U.; Köhler S. et al. (2016): Control and Evaluation Concept for smart MRO Approaches. *13th Global Conference on Sustainable Manufacturing - Decoupling Growth from Resource Use*, 700-705.

[Bru87] Brundenius C. (1987): Development and prospects of capital goods production in revolutionary Cuba. World Development. S. 95-112.

[Byr13] Byrne M.D. (2013): How Many Times Should a Stochastic Model Be Run? An Approach Based on Confidence Intervals. In: *Proceedings of the 12th International conference of cgnitive modeling*. Houston.

[Cook06] Cook M.; Bhamra T. und Lemon.M (2006): The transfer and application of product service systems: from academia to UK manufacturing firms. *Journal of Cleaner Production*, 14, 1455-1465.

[Dav10] Brink D. (2010): Statistical hypothesis testing. In *Essentials of Statistics*. S. 26-30;50-59.

[Den16] Denkena B.; Dittrich M.A. und Georgiadis A. (2016): Combining in-house pooling and sequencing for product regeneration by means of event-driven simulation. In: *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering*..

[ElM06] HA E. (2006): Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Manufacturing Systems*, 17, 261-276.

[Fin16] *Finite Elements Analysis*. Autodesk. http://www.autodesk.com/solutions/finite-element-analysis, zuletzt geprüft 2016.

[Geo14] Georgiadis A. (2014): Simulation-based Sequencing Algorithm for MRO Operations of Train Couplings. *Advanced Material Research*, 581-588.

[Gha16] Ghaddar B.; Sakr N. und Asiedu Y. (2016): Spare parts stockings analysis using genetic programming. *European Journal of Operational Research*, 136-144.

[Gho02] Ghobbar A.A. und Friend C.H. (2002): Sources of intermittent demand for aircraft spare parts within airline operations. *Journal Air Transportation Management*, 8, 4, 221-231.

[Gra99] Graves S.C. (1999): *Manufacturing Planning and Control*. Massachusetts Institute of Technology.

[Hees15] Hees A. und Reinhart G. (2015): Approach ofr production planning in reconfigurable manufacturing systems. *Procedia CIRP*, 33, 70-75.

[Hoa07] Hoad K.; Robinson S.; Davies R. et al. (2007): Automating des Output Analysis: How many replications to run. *Proceedings of the 2007 Winter Simulation Conference*.

[Jing15] Gu J.; Zhang G. und Li K.W. (2016): Efficient aircraft spare parts inventory management under demand uncertainty. *Journal of Air Transport Management*, 42, 101-109.

[Kelle14] Kellenbrink C.; Herde F.; Eickemeyer S.C. et al. (2014): Planning the regeneration processes of complex capital goods.

[Kis06] Kister T. und Hawkins B. (2006): Historical View of Maintenance. In *Maintenance Planning and Scheduling: Streamline Your Organization for a Lean Environment*. S. 1-18.

[Kist06] Kister T.C. und Hawkins B. (2006): Governing Principles and Concepts of Lean Maintenance. In *Maintenance Planning and Scheduling Handbook*. 1st Edition. Aufl. Butterworth-Heinemann. S. 42-62.

[Lau01] Swanson L. (2001): Linking maintenance strategies to performance. *International journal of production economics*.

[Lea16] *LeanManufacture*. http://www.leanmanufacture.net/, zuletzt geprüft am 15.July.2016.

[Lee08] Lee L.H.; Chew E.P. und Chen Y. (2008): Multi-objective simulation-based evolutionary algorithm for an aircraft spare parts allocation problem. *European Journal of Operational Research*, 189, 476-491.

[Men07] Menger C. (2007): Economy and Economics Goods. In *Principles of Economics*. Auburn: Ludwig von Mises Institute. S. 77-113.

[More03] Morelli M. (2003): Product-service systems, a perspective shift for designers: a case study: the design of a telecentre. *Design Studies*, 24, 73-79.

[Nyhu05] Nyhuis P.; von Ciemiski G. und Fischer A. (2005): Applying Simulation and Analytical Models for Logistics Performance Prediction. *Annals of the CIRP*, 54, 417-422.

[Nyhu06] Nyhuis P. (2006): Logistic Production Operating Curves - Basic Model of the Theory of Logistic Operating Curves. *Annals of the CIRP*, 55, 441-444.

[Reg05] Regattieri A.; Gamberi M.; Gamberini R. et al. (2005): Managing lumpy demand for aircraft spare parts. *Journal of Air Transport Management*, 11, 6, 426-431.

[Rob64] Robinson S. (1964): *Simulation: The Practice of Model Development and Use*. John Wiley & Sons, Ltd.

[Ros03] Rosenberg N. (2003): Capital goods, technology and economic growth. *Oxford Journals*, 217-227.

[RosD16] Rossetti M.D. (2016): Spreadsheet Simulation. In Wiley (Hg.), *Simulation Modeling and Arena*. 2nd. Aufl. New Jersey: John Wiley & Sons, Inc. S. 64-95.

[Ross10] Rossetti M.D. (2010): *Simulation Modeling and Arena*. Wiley.

[Ross16] Rossetti M.D. (2016): Analyzing Simulation Output. In Wiley (Hg.), *Simulation Modeling and Arena*. 2nd. Aufl. John Wiley & Sons, Inc. S. 300-384.

[SHAY13] A. Shayib M. (2013): *Applied Statistics*. 1st Edition. Aufl..

[Ste13] Steffen C. Eickemeyer, Tim Borcherding, Sebastian Schäfer, Peter Nyhuis (2013): Validation of data fusion as a method for forecastig the regeneration workload for complex capital goods. *German Academic Society for Production Engineering (WGP)*.

[Swi12] Swinney R. (2012): Inventory Pooling with Strategic Consumers: Operational and Behavioral Benefits.

[Vil07] Viles E.; Puente D.; Álvarez M.J. et al. (2007): Improving the corrective maintenance of an electronic system for trains. *Journal of Quality in Maintenance Engineering*, 13, 75-87.

[VRS12] V.R.S. Raju O.P.G.a.S.G.D. (2012): Maintenance, Repair and Overhaul Performance Indicators for Military Aircrafts. *Defence Science Journal*.

[Wan14] Wanke P. (2014): A Conceptual Framework for Inventory Management. *Production and Inventory Management Journal*, 49, 1, 6-23.

[Wel99] Welp E.G. und Endebrock K. (1999): Design for recyclability of capital goods. *Environmentally Conscious Design and Inverse Manufacturing*, 784-789.

[Wil15] Willem van Jaarsveld T.D.R.D. (2015): Improving spare parts inventory control at a repair shop. *Omega* , 57, 217-229.

[Wil96] Williams E.J. und Ahitov I. (96): Scheduling Analysis Using Discrete Event Simulation. *Proceeedings of Simulation*.

[Won07] Wong H.; van Oudheusden D. und Cattrysse D. (2007): Cost allocation in spare parts inventory pooling. *Transportation Research Part E: Logistics and Transportation Review*, 43, 4, 370-386.

[Xie11] Xie W. (2011): *Metaheuristics for single and multiple objectives producition scheduling for the capital goods industry*. Newcastle : Newcastle University Business School.

[Zei00] Zeigler B.P. (2000): Theory of Modellign and Simulation. *Academic Press Arizona*.

[Zhu12] Zhu H.; Gao J. und Tang D. (2012): A Web-based Product Service System for aerospace maintenance, repair and overhaul services. *Computers in Industry*, 63, 338-348.