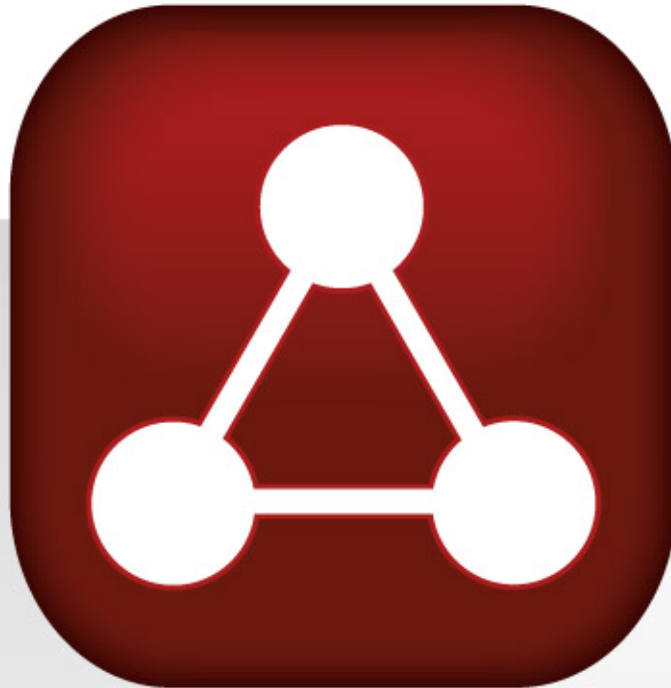


# Analytical Software Design Case

Nanda Technologies IM1000 Wafer Inspection System

Author: Rutger van Beusekom  
Version: 3.0 17/12/08



The simple way to build  
complex software  
systems

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

## Abstract

In this Analytical Software Design (ASD) case, Verum & Nanda Technologies used ASD to complete the design and development of the supervisory machine controller (SMC) software for NandaTech's IM1000 single wafer measurement system. The goals of the project were to:

- Complete the design and production of the SMC software in 8 weeks duration using on average 1 FTE for a budget of €32,000,-.
- Agree and model interfaces to 3rd party production integration software, to the hardware, and to the local machine's GUI and producing the control software between these interfaces.

ASD was used to:

- Verify the requirements for the Supervisory Machine Control, SMC software
- Model and mathematically verify all software interfaces, both external and internal
- Provide complete and correct interface specifications to (software) component suppliers
- Model and mathematically verify the SMC software
- Generate C++ source code directly from the verified SMC software models
- Provide the SMC software with a software infrastructure and operating framework.

The project results can be summarized as follows:

- Verum completed and mathematically verified the SMC software design and produced around 25,000 executable lines of code (elocs) of C++

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

- Productivity was approximately 80 elocs per man hour over the whole project
- During integration no errors were found in any ASD developed component
- ASD interface specifications (& simulations) were delivered to and used by the 3rd party software supplier and the hardware (PLC) supplier
- The project was completed exactly on time with exactly the estimated effort
- At delivery €29,000 of the agreed budget had been used. The remaining €3,000,- of the budget is reserved to help NandaTech with system integration when the HW is delivered

In conclusion the application of ASD resulted in the defect free development and delivery of the SMC software within the extremely tight budget and time constraints that NandaTech faced.

NandaTech concluded:

*“The only problems we found during the integration originated from the code written by hand the last three days; no problems were found in the ASD generated code.”*

**Raghuveera Ravinutala**, system architect at Nanda-Tech

*“I have taken part in a lot of software integration projects; none of them were as smooth as this one.”*

**Andreas Sittinger**, principal system engineer at Nanda-Tech

As a result of this project NandaTech became a Verum ASD:Suite customer.

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

## 1 Introduction

Nanda Technologies (Nanda) are developing software and hardware for quality control systems used in the semiconductor manufacturing industry. Nanda's high throughput macro defect inspection system provides a cost effective solution to increase overall fab yield.

Based on innovative technology Nanda is able to analyze a wafer at high speed, resulting in minimized feedback time and potentially significantly increased yield due to rework or early wafer scrapping.

By using state of the art imaging sensors and multiple illumination modes Nanda's inspection system is able to extract defects on a substrate with high fidelity and in real-time. Unique pattern recognition software allows the build-up of defect clusters over time to be monitored and to perform root-cause analysis.

The inspection module can be integrated directly into different semiconductor processing equipment to monitor every step of the manufacturing process. By analysing defect maps and process parameters, Nanda's production monitoring system is an important building block for advanced process control and automated process tool fault detection and correction. Developed on the Microsoft .NET platform and using the Interface A Standard, the system allows defect information to be shared throughout the entire production chain and enables immediate feedback to the process control system. It integrates easily into existing IT-infrastructures in semiconductor fabs as well as in other manufacturing environments where machine vision plays a critical role.

## 2 Technical Details

The IM1000 is a single wafer measurement system; wafers enter and leave the system via a single load/unload port. The system can be operated both as a standalone unit and one which is fully integrated into an automated wafer fab. It uses a motorised wafer stage to receive a wafer from an EFEM<sup>1</sup> or other automated wafer transport system.

---

<sup>1</sup> Equipment Front End Module

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

The system incorporates two different optical pathways to expose a wafer in order to highlight different defect characteristics. It uses a high-end digital inspection camera to acquire an image of an exposed wafer from which extract defect information is extracted and analysed.

The load/unload port door and the optical pathways are controlled by a PLC; both the PLC and the inspection camera are controlled from a PC. The entire system integrates into a 3rd party software framework, which controls the optional EFEM to deliver and remove the wafer, handles the wafer processing jobs and requests the IM1000 to process a previously loaded wafer.

Two distinguishing qualities of this system are its:

- defect recognition quality
- wafer throughput

In addition, as with all semiconductors manufacturing equipment, high uptime and continuous operation are essential.

Maximising wafer throughput requires prevention of delays resulting from unnecessary sequential operation of the steps to load, position, expose, analyse and unload a wafer. In other words parallelising operations and synchronising on their completion. Besides the concurrent operation, complexity is further increased by the need for the system to handle all possible hardware failures in order to achieve its operational reliability goals.

Because of this high level of complexity and desired reliability, and additional tight budget and schedule constraints, Nanda Technologies approached Verum to apply Analytical Software Design to the development of the IM1000 supervisory machine control software.

### 3 An overview of ASD

Analytical Software Design (ASD) is a proprietary technology developed by Verum; it is based on two design principles:

- Business critical software must be based on designs that are verified before implementation starts;

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

- Software Architects and Designers must use designs and architectures that can be verified using currently available tools and techniques.

With one exception, all branches of engineering routinely apply their specific branches of mathematics during specification and design. Modelling a design is cheaper than building a prototype and testing it. It is also more certain; testing is by definition an exercise in sampling and can never provide complete test coverage. There is no economically feasible amount of testing that can provide certainty of correctness. An architect charged with designing an earthquake-proof building does not build it and wait for an earthquake to test it! Instead, the design is mathematically modelled and subjected to rigorous mathematical analysis.

The one exception is software engineering. Apart from those few domains (mostly safety critical) where formal design and verification methods are mandated, software engineering mathematics is not routinely applied during software specification or design. Instead, reliance is placed on informal inspection-based methods and testing. As a consequence, defects injected early in the life-cycle during specification and design activities are frequently not detected and removed until after implementation is substantially complete and integration testing begins. This is the most expensive time to correct defects and occurs at a point in the life-cycle that results in the maximum impact on time to market. For most difficult kinds of errors, such as race conditions and deadlocks, this is also the least certain way to find them.

Analytical Software Design (ASD)<sup>2</sup> combines the practical application of software engineering mathematics and modelling with specification methods that avoid difficult mathematical notations and remain understandable to all project stakeholders. In addition, it uses statistical techniques for software component testing and advanced code generation techniques. From a single set of design specifications, the necessary mathematical models, program code and statistical test cases are generated automatically. ASD has been applied successfully to a number of industrial projects, the largest of which resulted in 300,000 executable lines of C++ program code.

---

<sup>2</sup> Patent applied for under patent application number GB 0410047.5

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

ASD uses the Sequence-based Specification method to specify functional performance requirements and designs as black box functions. These specifications are traceable to the original requirements specifications and remain completely accessible to the critical project stakeholders. This allows them to play a key role in verifying the ASD specifications and retain control over them. At the same time, ASD specifications provide the degree of rigor and precision necessary for mathematical analysis.

The ASD:ModelBuilder generates mathematical models which can be formally analyzed and verified using the ASD:ModelChecker. We can use the model checker to verify (i) whether a design satisfies its functional requirements; and (ii) whether the design uses other components according to their external functional specifications.

The ASD:CodeGenerator generates code automatically from the ASD specifications. The principle advantage of code generation is correctness; the code is generated automatically from the ASD specifications that have already been formally verified, resulting in significant efficiency gains.

When the design has been verified, the ASD:CodeGenerator is used to generate program source code in C++ or C# or other similar languages. The percentage of the total code that can be generated this way varies from project to project. Experience shows this is typically between 80% and 95%.

Finally, from the same set of design specifications, large numbers of statistically selected test cases can be generated in the form of self running tests and the results analysed to verify any manually written code.

## **4 Applying ASD to developing the IM1000**

Initially a working version of the software system architecture was derived in which functional components were identified and system functionality partitioned appropriately. Next, the requirements with respect to system behaviour were collected and documented by describing the major use

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

cases and the architecture was revised to include insights obtained while collecting the requirements. Figure 1 shows the resulting architecture diagram, the use of colours and shapes is explained below:

- The tan ellipses denote ASD component interfaces<sup>3</sup>
- The green ellipses denote plug-in interfaces
- The turquoise ellipses denote existing third party interfaces
- The white boxes denote ASD verified and generated components
- The yellow boxes denote handwritten stub implementations of one of the interface types above, with one exception, the HAL<sup>4</sup>.

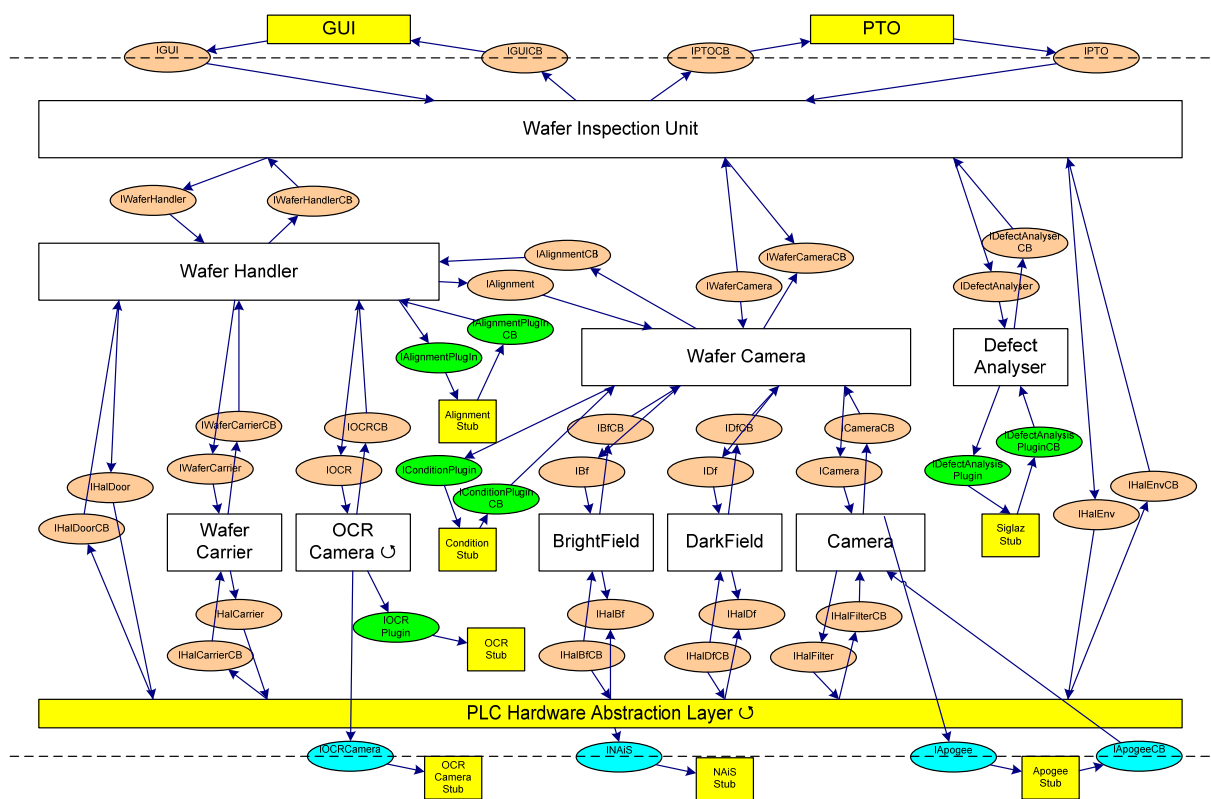


Figure 1: architecture of the IM1000 supervisory machine control

The system is partitioned in such a way that it:

- is reasonably well protected against future change
- allows for an acceptable design complexity per component

The part of the system to be built with ASD is chosen to be just above the PLC and Camera hardware drivers and below the client applications as indicated by the two dotted lines in Figure 1. Interfaces manage each

<sup>3</sup> an ASD interface always consists of a call to and a call back interface.

<sup>4</sup> the HAL is a layer that converts the ASD interfaces to read/writes on the PCL driver interface

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

boundary between ASD and the environment and between one ASD component and another.

At the start of the project, the design of the hardware and therefore the PLC software controlling it was not fully specified. This required defining a thin hardware abstraction layer to build abstract components on top. The interfaces with the naming convention IHal\* as well as IApogee\* span the bottom boundary. At the top, the 3rd party framework imposes an interface onto the system denoted by IPTO\*. To allow additional behaviour outside of the 3rd party framework an interface extension, denoted IGUI\*, is specified.

Next to the behavioural components, a number of functional<sup>5</sup> components were identified:

- Defect analysis
- Image conditioning
- Alignment feature detection
- Recipe parsing

These components were not designed with ASD because they are strictly algorithmic in nature and capture NandaTech's expertise. ASD was used to create the correct interface specifications and provide an initial implementation prior to these components being completed

To reduce development time, the system integrates with an existing 3rd party framework, which adds automated wafer handling and host connectivity; this then enables the system to be integrated it into a fully automated wafer fab. To develop a system that successfully integrates with such a framework, a clear and unambiguous understanding of its interactions is required. This required a visit to the Canadian supplier to model the interface using ASD. This specification serves two purposes: (i) it allows mathematical verification<sup>6</sup> of the IM1000 design to ensure that it behaves as expected and required by the framework; and (ii) it allowed an initial implementation to be generated that behaved according to this interface specification, which could then be used by the supplier to verify the specification by running it against their unit tests and acceptance tests in a very early stage of development.

---

<sup>5</sup> Components that do not keep state nor contribute to the system state space

<sup>6</sup> Verification asserts the absence of dead-lock and live-lock, it also asserts that the interface is completely and correctly implemented, it will not induce any illegal behaviour

## 5 Results

In the first week, a visit to Nanda resulted in the definition of the major use cases and the specification of the interfaces to the hardware. The hardware interface was specified using 55 canonical sequences<sup>7</sup> and 542 transition rules. In the second week, the 3rd party supplier was visited to specify the behaviour of an abstract process module as expected by the framework, which resulted in 28 canonical sequences and 815 transition rules.

The specification and design of the ASD interfaces and components was completed in 5 weeks, the mathematical verification of the designs took about 1 week. All drivers and stubs were implemented and integrated in 1 week. The remainder of the time was spent on meetings and travelling.

The main results of the project can be summarized as follows:

### Goals:

- 1) Verum originally proposed to complete the design and production of the software in 8 weeks duration using on average 1 FTE. The agreed budget was €32,000,-.
- 2) The project involved agreeing and modelling interfaces to 3rd party production integration software to PLC and Camera hardware, and to the local machine's GUI and producing the control software between these interfaces.

### Results:

- 1) Verum completed and mathematically verified the software design and produced around 25,000 lines (elocs) of C++.
- 2) Approximately 23,000 elocs were automatically generated from ASD models, 2000 elocs have been written by hand.

---

<sup>7</sup> A canonical sequence is an irreducible trace of events, in terms of a state diagram, a canonical sequence is identical to a state and each element in the sequence corresponds to a state transition.

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

- 3) In ASD terms the resultant design is relatively simple: 32,000 execution scenarios were mathematically verified (equivalent to 32,000 unique tests for full test coverage).
- 4) Productivity was approximately 80 elocs per man hour over the whole project. This compares to an industry standard of around 4 elocs per man hour.
- 5) The complete software was integrated and tested with the hardware and other components at Nanda-Tech. During integration no errors were found in any ASD developed component. This was the first time the software had been confronted with the hardware.
- 6) ASD interface specifications (& simulations) were delivered to and used by the 3rd party software supplier and the hardware (PLC) supplier.
- 7) The project has been completed exactly on time with exactly the estimated effort.
- 8) Verum performed extra work on the project including helping NandaTech sort out various problems with the HAL.
- 9) At delivery €29,000 of the agreed budget had been used. The remaining €3000,- of the budget is reserved to help NandaTech with system integration when the HW is delivered.
- 10) At completion, Nanda received a mathematically verified implementation completely documented by the accompanying sequence based specifications of the component interfaces and designs.
- 11) Modelling the interfaces with the environment in the earliest stage of the project allowed the development of the supervisory machine control software to be decoupled completely from all other parties involved.

# Verum white paper study

Analytical Software Design Case  
Nanda Technologies IM1000 Wafer Inspection System

## 6 Conclusions

The following conclusions were reached:

- Applying ASD allowed Nanda to meet their extremely tight schedule and budget constraints.
- ASD increased productivity as well as perceived quality.
- The software developed with ASD was error free.
- As a result of this project NandaTech became a Verum ASD:Suite customer.

## 8 Acknowledgements

We are grateful to Nanda Technologies for allowing us to present this case and for their cooperation when we applied these techniques to develop control software for the IM1000 wafer inspection system.