# XOOTIC magazine

July 2005-Volume 11-Number 1

*POST-MASTERS PROGRAMME SOFTWARE TECHNOLOGY*

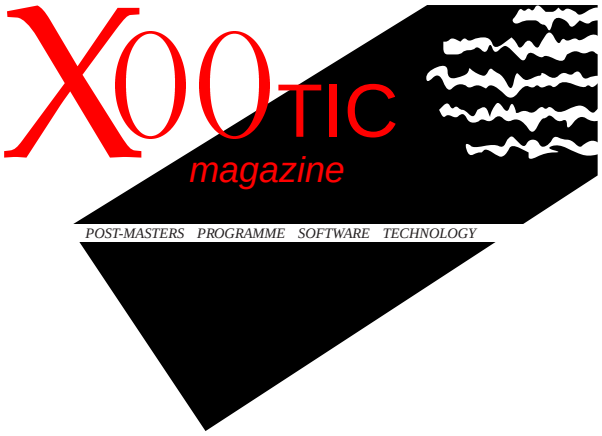
# GIS & ASD

## Contents



## Advertorials






## Colofon

XOOTIC MAGAZINE
Volume 11, Number 1
July 2005

*Editors*
S. Estok
M.M. Lindwer
M.P.W.J. van Osch
L. Posta

*Address*
XOOTIC and XOOTIC MAGAZINE
P.O. Box 6122
5600 MB Eindhoven
The Netherlands
xootic@win.tue.nl
`http://www.win.tue.nl/xootic/`

*Secretariat* OOTI
Post-masters Programme
Software Technology
Eindhoven University of Technology, HG 6.57
P.O. Box 513
5600 MB Eindhoven
The Netherlands
tel. +31 40 2474334
fax. +31 40 2475895
ooti@win.tue.nl
`http://wwwooti.win.tue.nl/`

*Printer*
Offsetdrukkerij De Witte, Veldhoven

# GIS & ASD

Editorial Preface

It's been a while since the last issue of the XOOTIC Magazine appeared. A lot has happened since then. There were changes in the XOOTIC board and in other XOOTIC committees. A major change happened in the XOOTIC Magazine Committee. The editorial board in 2005 consists of Menno Lindwer, Slavomir Estok, Michiel van Osch, and Ladislau Posta. From this place we would like to thank Nico Kuijpers, Yarema Mazuryk, and Chris Delnooz for their work as editors of the XOOTIC Magazine.

Due to various reasons, the XOOTIC Magazine Committee decided to not be bounded by the "one theme issue" for the magazine. Therefore, the current issue contains articles from the fields of Geographic Information Systems (GIS) and Analytical Software Design (ASD).

In the first article, Marc van Kreveld and Bettina Speckmann present the first algorithms for rectangular cartogram construction. A rectangular cartogram is a type of map where every region is a rectangle. The size of the rectangles is chosen such that their areas represent a geographic variable (e.g. population).

In the second article, Rob Dekker reports the results of an investigation of several object-based classifications and change detection techniques used in map updating.

Last, but not least, Guy Broadfoot and George Kielty, present how Verum applied ASD, a new approach that applies software engineering mathematics to industrial software development, to develop the software controlling an advanced mechatronics subsystem being developed by Philips Applied Technologies, Department of Mechatronics, in the Netherlands.

The magazine ends with a list of recent OOTI publications.

Enjoy reading this magazine!

Ladislau Posta, editor

# Toekomst in een baan bij Topic

**De toekomst heb je grotendeels zelf in de hand. Ook als het om je loopbaan gaat. Daarvoor ben je bij TOPIC zelf verantwoordelijk. Uiteraard staan we je hierin bij. Bij TOPIC werk je aan de meest uitdagende klussen. Topklussen, waardoor je sneller doorgroeit.**

Wat wil je bereiken? Dat bepaal jij zelf. Om je te helpen bij het vaststellen van je doelen en de realisatie hiervan, voer je bij TOPIC een voorjaarsgesprek. Dit gesprek gaat uit van jouw wensen en ambities. Hierin kun je aangeven welke stap in je carrière je wilt zetten, wat je wilt bereiken op korte en op lange termijn en hoe je bij je gestelde doelen wilt komen. Het gesprek wordt besloten met een handtekening van beide zijden. Klinkt formeel? Zeker, maar dat is een bewuste keuze. Zo zijn jij en TOPIC er zeker van dat er met de afspraken serieus wordt omgegaan. Je toekomst verwezenlijk je dus bij TOPIC.

**TOPIC groeit hard, goede mensen zijn welkom**

TOPIC
SOFTWARE GROUP
EMBEDDED IN YOUR FUTURE

Topic Software Group, postbus 440, 5680 AK Best, tel 0499 33 69 79, e-mail info@topic.nl , internet www.topic.nl

# Automated Construction of Rectangular Cartograms[1]

Marc van Kreveld, Bettina Speckmann

*A rectangular cartogram is a type of map where every region is a rectangle. The size of the rectangles is chosen such that their areas represent a geographic variable (e.g., population). Good rectangular cartograms are hard to generate: The area specifications for each rectangle may make it impossible to realize correct adjacencies between the regions and so hamper the intuitive understanding of the map.*

*We present the first algorithms for rectangular cartogram construction. Our algorithms depend on a precise formalization of region adjacencies and build upon existing VLSI layout algorithms. An implementation of our algorithms and various tests show that in practice, visually pleasing rectangular cartograms with small cartographic error can be generated effectively.*

## Introduction

Cartographers have developed many different techniques to visualize statistical data about a set of regions like countries, states or counties. *Cartograms* are among the most well known and widely used of these techniques. The regions of a cartogram are deformed such that the area of a region corresponds to a particular geographic variable [3]. The most common variable is population: In a population cartogram, the areas of the regions are proportional to their population. Since the sizes of the regions are not their true sizes they generally cannot keep both their shape and their adjacencies. A good cartogram, however, preserves the recognizability in some way.

Globally speaking, there are three types of cartogram. The standard type (the *contiguous area cartogram*) has deformed regions so that the desired sizes can be obtained and the adjacencies kept. Algorithms for such cartograms are described in [4, 5, 8, 15]. The second type of cartogram is the non-contiguous area cartogram [12]. The regions have the true shape, but are scaled down and generally do not touch anymore. The third type of cartogram is the rectangular cartogram, introduced by Raisz in 1934 [13], where each region is represented by a rectangle. This has the advantage that the sizes (area) of the regions can be estimated much better than with the first two types.

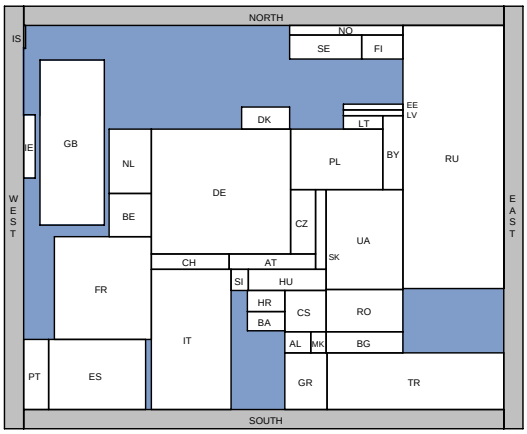



Figure 1: The population of Europe (country codes according to the ISO 3611 standard).

Algorithms for cartograms have been studied for over thirty years, but no method for producing rectangular cartograms has been developed so far [16].

[1] This work has previously been published as part of [11].

**Quality criteria.** Whether a rectangular cartogram is good is determined by several factors. One of these is the *cartographic error* [4, 5], which is defined for each region as $| A_c - A_s | / A_s$, where $A_c$ is the area of the region in the cartogram and $A_s$ is the specified area of that region, given by the geographic variable to be shown. The following list summarizes the most important quality criteria:

- Average cartographic error.
- Maximum cartographic error.
- Correct adjacencies of the rectangles.
- Maximum aspect ratio.
- Suitable relative positions.

For a purely rectangular cartogram we cannot expect to simultaneously satisfy all criteria well. Recently, Heilmann et al. [6] presented rectangular map approximations that have zero cartographic error but do not satisfy the other criteria.

**Related work.** Rectangular cartograms are closely related to *floor plans* for electronic chips. Floor planning aims to represent a planar graph by its *rectangular dual*, defined as follows. A *rectangular partition* of a rectangle $R$ is a partition of $R$ into a set $\mathcal{R}$ of non-overlapping rectangles such that no four rectangles in $\mathcal{R}$ meet at the same point. A rectangular dual of a planar graph $(G, V)$ is a rectangular partition $\mathcal{R}$, such that $(i)$ there is a one-to-one correspondence between the rectangles in $\mathcal{R}$ and the nodes in $G$, and $(ii)$ two rectangles in $\mathcal{R}$ share a common boundary if and only if the corresponding nodes in $G$ are connected. The following theorem was proven in [10]:

**Theorem 0.1** *A planar graph G has a rectangular dual R with four rectangles on the boundary of R if and only if*

1. *every interior face is a triangle and the exterior face is a quadrangle, and*
2. *G has no separating triangles.*

Most maps give rise to triangulated graphs, because usually at most three regions meet at any one point. Separating triangles occasionally arise, for example, Luxembourg does not border any sea and is incident to only three countries. This implies that a purely rectangular cartogram with correct adjacencies does not exist for Europe. Also note that although every triangulated planar graph without separating triangles has a rectangular dual this does not imply that an error free cartogram for this graph exists.

**Results.** We present the first fully automated algorithms for the computation of rectangular cartograms. We formalize the region adjacencies based on their geographic location and are so able to enumerate and process all feasible *rectangular layouts* for a particular subdivision (i.e., map). The precise steps that lead us from the input data to an algorithmically processable rectangular subdivision are sketched in Section .

In [11] we describe three algorithms that compute a cartogram from a rectangular layout. Here we concentrate on the simplest one of these, the so-called *segment moving heuristic*. We evaluated this easy and efficient heuristic experimentally. The results of our implementation can be found in Section . A Java prototype can be seen at `http://www.win.tue.nl/~speckman/demos/carto`.

## Algorithmic Outline

Assume that we are given an administrative subdivision into a set of regions. The adjacencies of the regions can be represented in a graph $F$, which is the face graph of the subdivision.

**1. Preprocessing:** The face graph $F$ is in most cases already triangulated (except for its outer face). In order to construct a rectangular dual of $F$ we first have to process internal vertices of degree less than four and then triangulate any remaining non-triangular faces.

**2. Directed edge labels:** Any two nodes in the face graph have at least one direction of adjacency which follows naturally from their geographic location. While in theory there are four different directions of adjacency any two nodes can have, in practice only one or two directions are reasonable.

Our algorithms go through all possible combinations of direction assignments and determine which one gives a correct or the best result. While in theory there can be an exponential number of options, in practice there is often only one natural choice for the direction of adjacency between two regions. We call a particular choice of adjacency directions a *di-*

*rected edge labeling*. A face graph $F$ with a directed edge labeling can be represented by a rectangular dual if and only if

1. every internal region has at least one North, one South, one East, and one West neighbor, and
2. when traversing the neighbors of a node in clockwise order starting at the western most North neighbor we first encounter all North neighbors, then all East neighbors, then all South neighbors and finally all West neighbors.

A realizable directed edge labeling constitutes a *regular edge labeling* for $F$ as defined in [7] which immediately implies our observation.

**3. Rectangular layout:** To actually represent a face graph together with a realizable directed edge labeling as a rectangular dual we have to pay special attention to the nodes on the outer face since they may miss neighbors in up to three directions. To compensate for that we add four special regions NORTH, EAST, SOUTH, and WEST, as well as *sea regions* that help to preserve the original outline of the subdivision. Then we can employ the algorithm by He and Kant [7] to construct a *rectangular layout*, i.e., the unique rectangular dual of a realizable directed edge labeling. The output of our implementation of the algorithm by He and Kant is shown in Figure 2.

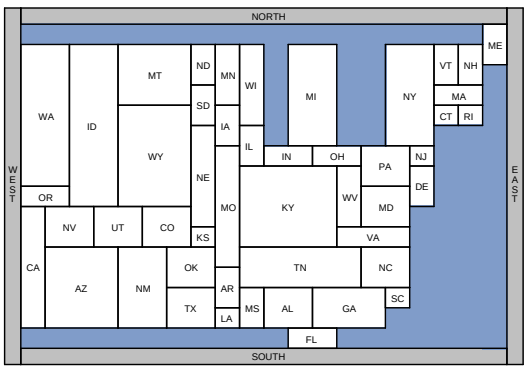



Figure 2: One of 4608 possible rectangular layouts of the US.

**4. Area assignment:** For a given set of area values and a given rectangular layout we would like to decide if an assignment of the area values to the regions is possible without destroying the correct adjacencies. Should the answer be negative or should the question be undecidable, then we still want to compute a cartogram that has a small cartographic error while maintaining reasonable aspect ratios and relative positions.

**Segment moving heuristic.** A simple but efficient heuristic that works as follows. Consider the maximal vertical segments and maximal horizontal segments in the layout, for example the vertical segment in Figure 2 that has Kentucky (KY) to its left and West Virginia (WV) and Virginia (VA) to its right. This segment can be moved a little to the left, making Kentucky smaller and the Virginias larger, or it can be moved to the right with the opposite effect.

The segment moving heuristic loops over all maximal segments and moves each with a small step in the direction that decreases the maximum error of the adjacent regions. After a number of iterations, one can expect that all maximal segments have moved to a locally optimal position. However, we have no proof that the method reaches the global optimum, or that it even converges.

The segment moving heuristic has some important advantages: $(i)$ it can be used for any rectangular layout, $(ii)$ one iterative step for all maximal segments takes $O(n)$ time, $(iii)$ no area need to be specified for sea rectangles, $(iv)$ a bound on the aspect ratio can be specified, and $(v)$ adjacencies between the rectangles can be preserved, but need not be. Not preserving adjacencies can help to reduce cartographic error.

## Implementation and experiments

We have implemented the segment moving heuristic and tested it on several data sets. The main objective was to discover whether rectangular cartograms with reasonably small cartographic error exist, given that they are rather restrictive in the possibilities to represent all rectangle areas correctly. Obviously, we can only answer this question if the segment moving heuristic actually finds a good cartogram if it exist. Secondary objectives of the experiments are to determine to what extent the cartographic error depends on maximum aspect ratio and correct or false adjacencies. We were also interested in the dependency of the error on the percentage of area used by the sea.

Our layout data sets consist of the 36 countries of Europe and the 48 contiguous states of the USA. For

| Data set | Sea | Aspect ratio | Ave. error | Max. error |
|---|---|---|---|---|
| Eu elec. | 20% | 8 | 0.071 | 0.280 |
| Eu elec. | 20% | 9 | 0.070 | 0.183 |
| Eu elec. | 20% | 10 | 0.067 | 0.179 |
| Eu elec. | 20% | 11 | 0.065 | 0.155 |
| Eu elec. | 20% | 12 | 0.054 | 0.137 |
| Eu elec. | 10% | 10 | 0.098 | 0.320 |
| Eu elec. | 15% | 10 | 0.076 | 0.245 |
| Eu elec. | 20% | 10 | 0.067 | 0.179 |
| Eu elec. | 25% | 10 | 0.049 | 0.126 |

Table 1: Errors for different aspect ratios and sea percentages (correct adjacencies).

Europe, we joined Belgium and Luxembourg, and Ukraine and Moldova, because rectangular duals do not exist if Luxembourg or Moldova are included as a separate country. Europe has 16 sea rectangles and the US data set has 9. For Europe we allowed 10 pairs of adjacent countries to be in different relative position, leading to 1024 possible layouts. Of these, 768 correspond to a realizable directed edge labeling. For the USA we have 13 pairs, 8192 possible layouts, and 4608 of these are realizable. In the experiments, all 768 or 4608 layouts are considered and the one giving the lowest average error is chosen as the cartogram.

As numeric data we considered for Europe the *population* and the *electricity production*, taken from [2]. For the USA we considered *population*, *native population*, number of *farms*, and total length of *highways*. The data is provided by the US census bureau in the *Statistical Abstract of the United States*.[1]

Preliminary tests on all data sets showed that the false adjacency option always gives considerably lower error than correct adjacencies. The false adjacency option always allowed cartograms with average error of only a few percent. A small part of the errors is due to the discrete steps taken when moving the segments. Since cartograms are interpreted visually and show a global picture, errors of a few percent on the average are acceptable. Errors of a few percent are also present in standard, computer-generated contiguous cartograms [4, 5, 8, 9]. We note that most hand-made rectangular cartograms also have false adjacencies and that aspect ratios of more than 20 can be observed.

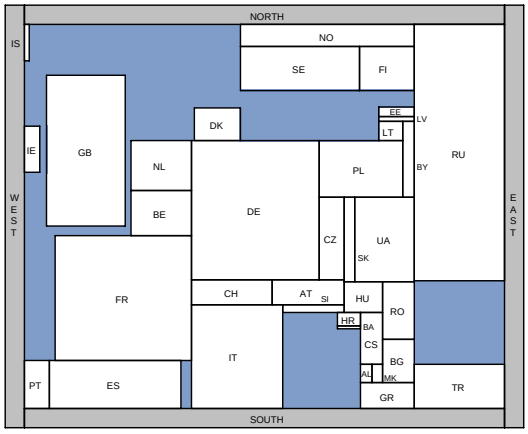



Figure 3: A cartogram depicting the electricity production of Europe.

Table 1 shows errors for various settings for the electricity production data set. The rectangular layout chosen for the table is the one with lowest average error. The corresponding maximum error is shown only for completeness. In the table we observe that the error goes down with a larger allowed aspect ratio, as expected. For Europe and population (not shown in the table), errors below 0.1 on the average with correct adjacencies were only obtained for aspect ratios greater than 15. The table also shows that a larger sea percentage brings the error down. This is as expected because sea rectangles can grow or shrink to reduce the error of adjacent countries, while a sea rectangle cannot have an error in its area. So, more sea means more freedom to reduce errors. However, sea rectangles should not become so small that they visually (nearly) disappear.

Table 2 shows errors for various settings for two US data sets. Again, we choose the rectangular layout giving the lowest average error. In the US highway

[1] `http://www.census.gov/statab/www/`

| Data set | Adjacency | Aspect ratio | Ave. error | Max. error |
|---|---|---|---|---|
| US population | false | 8 | 0.104 | 0.278 |
| US population | false | 9 | 0.085 | 0.193 |
| US population | false | 10 | 0.052 | 0.295 |
| US population | false | 11 | 0.030 | 0.091 |
| US population | false | 12 | 0.022 | 0.056 |
| US population | correct | 12 | 0.327 | 0.618 |
| US population | correct | 13 | 0.319 | 0.608 |
| US population | correct | 14 | 0.317 | 0.612 |
| US population | correct | 15 | 0.314 | 0.569 |
| US population | correct | 16 | 0.308 | 0.612 |
| US highway | correct | 6 | 0.073 | 0.188 |
| US highway | correct | 7 | 0.059 | 0.111 |
| US highway | correct | 8 | 0.058 | 0.101 |
| US highway | correct | 9 | 0.058 | 0.101 |
| US highway | correct | 10 | 0.058 | 0.101 |

Table 2: Errors for different aspect ratios, and correct or false adjacencies. Sea 20%.

data set, aspect ratios above $8$ do not seem to decrease the error below a certain value. In the US population data set, correct adjacencies give a larger error than is acceptable. Even an aspect ratio of $40$ gave an average error of over $0.3$. We ran the same tests for the native population data and again observed that the error decreases with larger aspect ratio. An aspect ratio of 7 combined with false adjacency gives a cartogram with average error below $0.04$ (see Fig. 4). Only the highways data allowed correct adjacencies, small aspect ratio, and small error simultaneously.

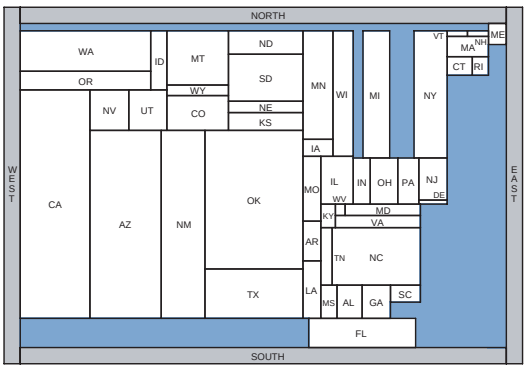



Figure 4: A cartogram depicting the native population of the United States.

Figures 4, 5, 6, and 7 show rectangular cartograms of the US. Three of them have false adjacencies, but we can observe that adjacencies are only slightly disturbed in all cases (which is the same as for hand-made rectangular cartograms). The data sets allowed an aspect ratio of $10$ or lower to yield an average error between $0.03$ and $0.06$, except for the farms data. Here an aspect ratio of $20$ gives an average error of just below $0.1$. Figures 1 and 3 show rectangular cartograms for Europe. The former has false adjacencies and aspect ratio bounded by $12$, the latter has correct adjacencies and aspect ratio bounded by $8$. The average error is roughly $0.06$ in both cartograms.

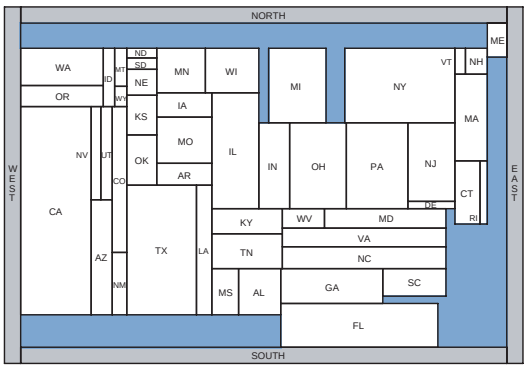



Figure 5: The population of the US.

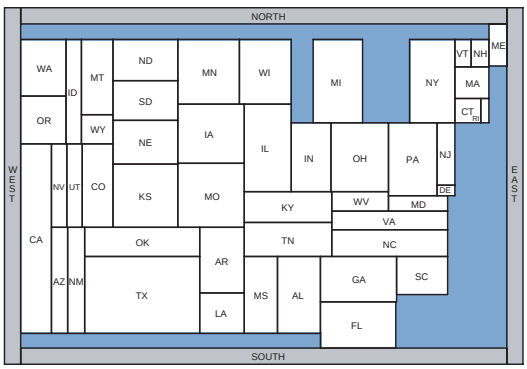



Figure 6: The highway kilometers of the US.

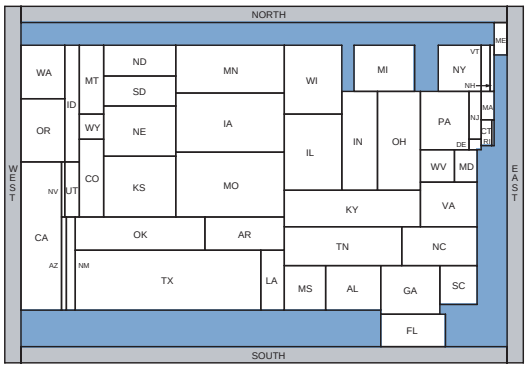



Figure 7: The farms of the US.

## Conclusion

In this paper we presented the first algorithms to compute rectangular cartograms. We showed how to formalize region adjacencies in order to generate algorithmically processable layouts. An interesting open problem is whether rectangular cartogram construction (correct or minimum error) can be done in polynomial time.

We experimentally studied the quality of our segment moving heuristic and showed that it is very effective in producing aesthetic rectangular cartograms with only small cartographic error. Our tests show the dependency of the error on the aspect ratio, correct adjacencies, and sea percentage. The quality of the cartograms generated is comparable to hand-made rectangular cartograms.

## References


[1] M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear Programming - Theory and Algorithms*. John Wiley & Sons, Hoboken, NJ, 2nd edition, 1993.

[2] *De Grote Bosatlas*. Wolters-Noordhoff, Groningen, 52nd edition, 2001.

[3] B. Dent. *Cartography - thematic map design*. McGraw-Hill, 5th edition, 1999.

[4] J. A. Dougenik, N. R. Chrisman, and D. R. Niemeyer. An algorithm to construct continous area cartograms. *Professional Geographer*, 37:75–81, 1985.

[5] H. Edelsbrunner and E. Waupotitsch. A combinatorial approach to cartograms. *Comput. Geom. Theory Appl.*, 7:343–360, 1997.

[6] R. Heilmann, D. A. Keim, C. Panse, and M. Sips. Recmap: Rectangular map approximations. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS)*, pages 33–40, 2004.

[7] G. Kant and X. He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theoretical Computer Science*, 172:175–193, 1997.

[8] D. Keim, S. North, and C. Panse. Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Transactions on Visualization and Computer Graphics*, 10:95–110, 2004.

[9] C. Kocmoud and D. House. A constraint-based approach to constructing continuous cartograms. In *Proceedings of the Symposium on Spatial Data Handling*, pages 236–246, 1998.

[10] K. Koźmiński and E. Kinnen. Rectangular dual of planar graphs. *Networks*, 5:145–157, 1985.

[11] M. v. Kreveld and B. Speckmann. On rectangular cartograms. *Computational Geometry: Theory and Applications*, 2005. (to appear).

[12] J. Olson. Noncontiguous area cartograms. *The Professional Geographer*, 28:371–380, 1976.

[13] E. Raisz. The rectangular statistical cartogram. *Geographical Review*, 24:292–296, 1934.

[14] S. Sur-Kolay and B. Bhattacharya. The cycle structure of channel graphs in nonslicable floorplans and a unified algorithm for feasible routing order. In *Proceedings of the IEEE International Conference on Computer Design*, pages 524–527, 1991.

[15] W. Tobler. Pseudo-cartograms. *The American Cartographer*, 13:43–50, 1986.

[16] W. Tobler. Thirty-five years of computer cartograms. *Annals of the Association of American Geographers*, 94(1):58–71, 2004.

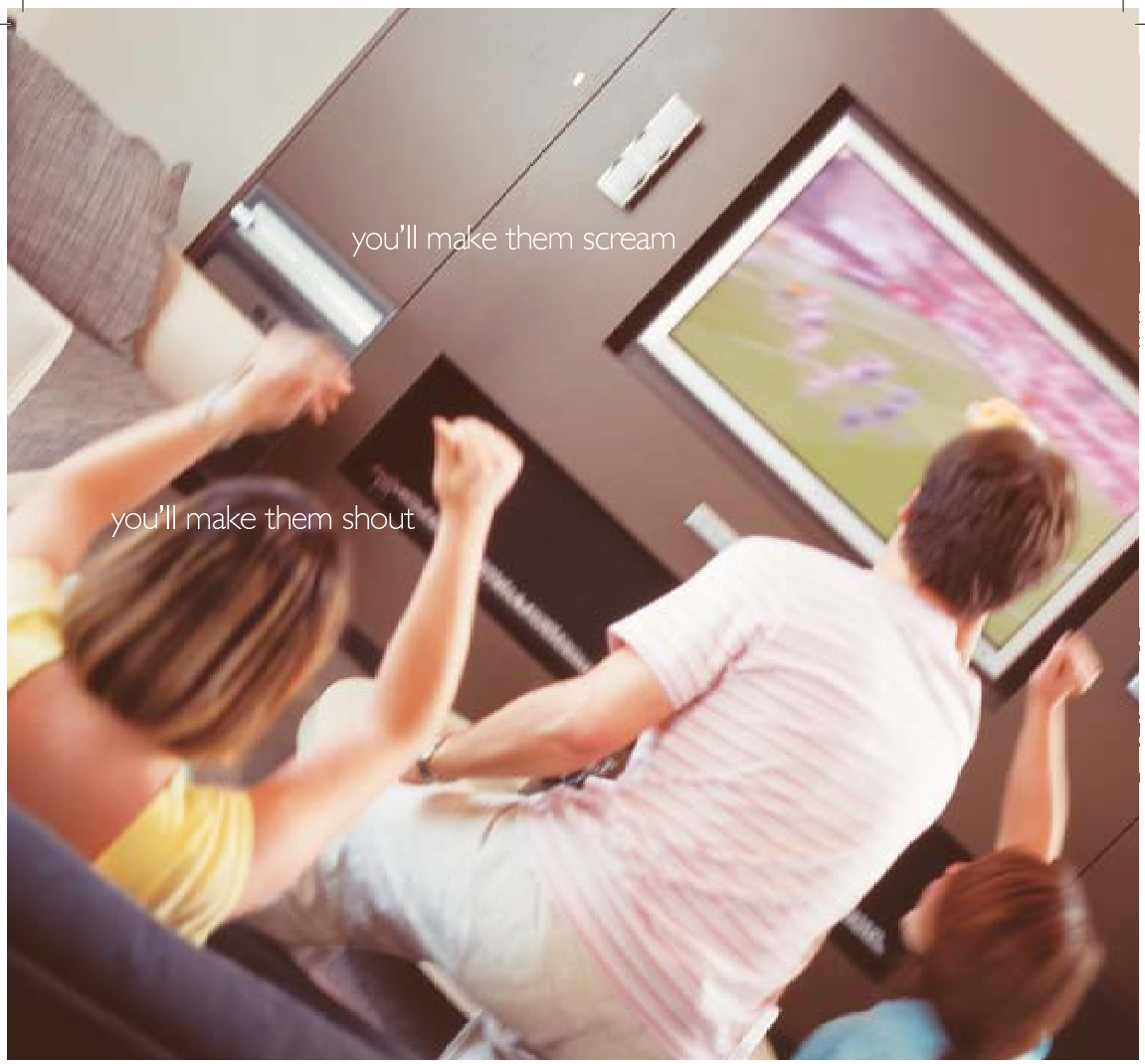
## Contact Information

**Marc van Kreveld**

Institute for Information and Computing Sciences
Utrecht University
P.O. Box 80.089 3508 TB Utrecht
The Netherlands
marc@cs.uu.nl

**Bettina Speckmann**

Department of Mathematics and Computer Science
TU Eindhoven
P.O. Box 513
5600 MB Eindhoven
The Netherlands
speckman@win.tue.nl




# touch lives every day

Elke dag weer komen miljoenen mensen over de hele wereld in aanraking met Philips producten. Of het nu gaat om de mobiele telefoon voor het uitwisselen van de laatste nieuwtjes, de imposante verlichting in een voetbalstadion of de televisie die mensen in spanning houdt. Philips wil het leven van mensen verbeteren en veraangenamen met geavanceerde en zinvolle technologie. Bij ons kunnen jouw gedachten en ideeën daar ook een bijdrage aan leveren. Bij Philips kun je het dagelijkse leven van mensen positief beïnvloeden.

## Technical Software Engineers • Philips TASS • Eindhoven

Philips TASS (Technical Application Software Services) is gespecialiseerd in de ontwikkeling van technische software voor onder meer elektronische producten, medische en communicatieapparatuur en industriële automatisering. Onze 250 software professionals in Eindhoven en Brussel zijn actief op gebieden als hardware/software interfacing, integratie & testen en remote services. Wij stellen onze expertise ter beschikking van bedrijven binnen en buiten Philips.

Wil jij als software engineer meewerken aan de producten van morgen, van consumentenapplicaties tot geavanceerde professionele toepassingen? En ben je op zoek naar een carrière als (senior) software architect of consultant? Dan biedt Philips TASS jou het beste van twee werelden: de informele sfeer en afwisselende werkomgeving van een softwarehuis, gecombineerd met alle kansen van een multinational. Wij vragen een relevante vooropleiding (minimaal HBO) en 2 tot 5 jaar werkervaring. Een drive voor technische software, een proactieve werkhouding en goede communicatieve vaardigheden zijn essentieel.

Als je jezelf herkent in deze beschrijving, nodigen wij je uit on-line te reageren via onze website **www.philips.nl/werken**. Gebruik referentie 19911.

**PHILIPS**
sense and simplicity

# Object-based updating of land-use maps of urban areas using satellite remote sensing

R.J. Dekker

*Geographical information in the form of maps is continuously subjected to change, especially in urban areas. Therefore maps have to be updated, which can be done using satellite remote sensing techniques since many satellites are in orbit today. In this paper several object-based classification and change detection techniques are investigated. An important aspect in map-updating is the translation from land cover (image domain) to land use (map domain). To study the results of several techniques, data of Landsat 5 TM (30 m), ERS 1 (30 m), Ikonos (4 m) and PHARUS (4 m) were used. The focus was on data of urban areas in the Netherlands. Classification of the images was done using spectral information, texture and in one case information on the relation between adjacent objects. Non-parametric techniques were applied because most textures appeared to be non-Gaussian distributed. The classification accuracy of Landsat 5 TM was best, second was ERS 1 and third were the results of PHARUS and Ikonos. Several reasons are given, but classifying high-resolution images is clearly more difficult than classifying low-resolution images. In case of change detection, pre-classification techniques were preferred. Although the methods can be improved, more information is required, e.g. from combining sensors or from the map to be updated. Map updating may not become fully automatic, but the job of a human operator can be made easier using the techniques investigated in this paper.*

## Introduction

The world is rich of geographical information in the form of maps. We all know paper maps but today, more and more maps become available digitally. Examples of such maps are the Digital Chart of the World (DCW), the Vector Map (VMap) product series and national digital maps as the TOPvector product series of the Topografische Dienst Kadaster, Netherlands national mapping agency. The areas of applications of those maps are various: environmental planning, agriculture, forestry, tourism, defence, and many more. Because the Earth's surface, that maps attempt to describe, develops, maps are continuously subjected to change, especially in urban areas where the pace of development is relatively high. To keep maps up to date, it is important to know where the change took place, what has changed, how it is changed and if it is relevant for the map. To answer these questions satellite remote sensing techniques can be used, since many commercial satellites are currently in orbit. Examples are Landsat, Spot, Ikonos, Quickbird, ERS, Envisat, and Radarsat. More are planned.

The remote sensing techniques that are discussed in this paper focus on land-use classification and change detection. An important development in these techniques that is addressed here is object orientation. For evaluation some of these techniques were applied to satellite data of several areas in the Netherlands, including data of Landsat 5 TM (30 m), ERS 1 (30 m), Ikonos (4 m) and PHARUS (4

m). Landsat 5 and Ikonos are optical/infrared satellites. ERS 1 is a radar satellite. PHARUS (Greidanus et al. 1996) is an airborne imaging radar representing the future generation of radar satellites which is planned to be in orbit from 2005 (e.g. Radarsat 2, Cosmo/SkyMed and TerraSAR). This paper is based on an earlier publication by the author ([6]).

## Land Cover and Land Use

An important aspect in translating satellite images into maps is that satellites give a physical description of the earth's surface (materials, surface roughness, structure), while most maps give a functional (socio-economic) description. Both descriptions are referred to as land cover and land use respectively and are often mixed up ([1], [9]). Examples of land cover are grass, trees, building and asphalt. Examples of land use are agricultural, residential, commercial and industrial. Most land use classes are composed of several land cover types, and have often a many-to-many relationship ([11], [4], [9]), see also figure 1.

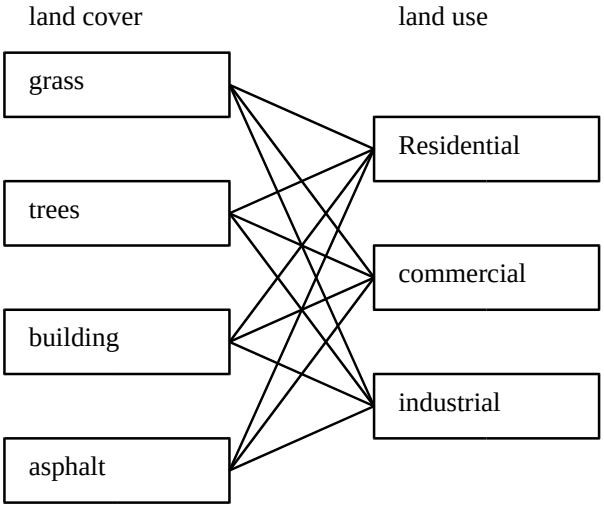



Figure 1: Many-to-many relationship between land cover and land use (Fisher et al. 2002).

## Test Areas and Data

In this paper classification and change detection is applied to satellite images of the Netherlands. The Netherlands can be characterised as a well-planned country: almost every acre has a function. Compared to many other countries the Netherlands is quite urbanised, especially in the west and south. Three test areas were selected:

- Zwolle and Veluwe, in the centre to the East of the Netherlands, is a less dense urban area. It is dominated by meadow and forest (Veluwe).
- Randstad Holland, a dispersed but dense urban area in the west of the Netherlands. It contains residential areas, industry, greenhouses, pasture, arable land, and some forest. The area includes two of the four largest cities in the Netherlands, Rotterdam and The Hague.
- The Hague, in the west of the Randstad Holland by the North Sea is quite a green city. It contains much forest and is built against a (natural) sand dune area that protects a large part of the Netherlands from the sea. This area is chosen to study higher resolution data.

The maps of the first two areas are from VMap level 1 of the Netherlands (1:250,000) that was produced in 1998. The information model behind VMap is the Digital Geographic Information Exchange Standard - Feature Attribute Coding Catalogue ([8]) which was developed by the Digital Geographic Information Working Group ([8]). About 150 types of entities are included. Because some VMap level 1 land-use types are rather close, the map was conceptually generalised by merging areas with different codes into a map with less classes. For instance, forest includes the classes orchard/plantation and trees, which is conceptually valid in the Netherlands. Unfortunately, VMap level 1 does not show areas that contain industrial activity only, if reproduced these are included in urban. Figure 2 shows the map of the area of Zwolle and Veluwe. The map of The Hague comes from the TOP10vector series of the Netherlands (1:10,000). It was updated in 1999. Due to the large number of land-use classes the TOP10vector was conceptually generalised as well. It is shown in Figure 3. The satellite data are summarised in table 1. There is no relation between useful resolution and pixel spacing (i.e. sampling distance). If the resolution is higher than the pixel spacing, this means that the images are oversampled. The number of looks is a measure for the speckle-noise reduction. For instance, when the number of looks is three, this means that the speckle-noise is reduced by averaging three independent images (the band-

width of one image corresponds to one-third of the antenna beam). The polarisation stands for the orientation of the radar waves. The backscatter properties of objects can be different for different polarisations. Some acquisitions are previous to the map updates but the differences in land use are small.

| **sensor** | **Landsat 5 TM** | **ERS 1** | **Ikonos** | **PHARUS** |
|---|---|---|---|---|
| type | optical/infrared | radar | optical/infrared | radar (airborne) |
| altitude | 705 km | 785 km | 681 km | 6 km |
| useful resolution | 30 m | 30 m | 4 m | 4 m |
| useful bands | 6 (0.45-2.35 $\mu$m) | 1 (5.3 cm) | 4 (0.45-0.88 $\mu$m) | 1 (5.3 cm) |
| visual | 3 | - | 3 | - |
| infrared | 3 | - | 1 | - |
| microwave | - | 1 | - | 1 |
| | | | | |
| number of images | 1 | 1 | 1 | 2 |
| number of looks | N/A | 3 | N/A | 5 |
| polarisation (used) | N/A | VV | N/A | HH |
| | | | | |
| test area | Zwolle and Veluwe | Randstad Holland | The Hague | The Hague |
| acquisition data | 18 October 1993 | 23 June 1995 | 23 September 2000 | 26 April 1996<br>27 January 1998 |
| pixel spacing | 20 m | 20 m | 2 m | 2 m |
| size | 1500x1500 pixels | 3209x3273 pixels | 1353x1825 pixels | 1353x1825 pixels |
| | 30x30 km | 64x65 km | 2.7x3.7 km | 2.7x3.7 km |

Table 1: Overview of satellite data of the three test areas.

## Classification Techniques

Classification techniques, how to translate a satellite image into land cover or even land use, can be divided in various categories. Two are described: feature-based and knowledge-based (i.e. rule-based) techniques. In feature-based classification images are classified based on a set of distinguishing characteristics or features. A feature can be the spectral intensity, texture, polarimetric information, etc. Texture can become important if only one spectral band is available, e.g. in case of the radar images. Therefore a set of texture measures was investigated with respect to their separability of land use in the ERS 1 image of the Randstad Holland ([6]). The measures that performed best were mean intensity (actually no texture), variance, weighted-rank fill ratio and semivariograms. The weighted-rank fill ratio is an order-statistic, which is defined as the ratio of the power of the 5% brightest pixels of an object and the total power of all pixels. The semivariogram is a eo-statistic measure, which is an indicator of the geospatial distribution of textures (e.g. repeatability, depth).

The most popular feature-based classifiers are parametric, meaning that they decide if a feature-vector belongs to a certain class, based on the parameters of the class probability-density-function (e.g. mean and standard deviation). A problem with these classifiers is that most assume the features to be normally distributed. Although other classifiers can be designed, there can still be the problem of features having different distributions. Lining up these distributions is sometimes possible by applying variable transformations, but not always. Another solution is to apply non-parametric techniques, which consider the whole population and not only the mean and standard deviation. An example of a non-parametric classifier is the *k*-nearest neighbour (*k*NN) classifier which is based on the following distance ([10]):

$$d_i^2 = (X - X_{i,NN})^T \sum_i^{-1} (X - X_{i,NN})$$

Here $X_{i,NN}$ is the *k*-th nearest neighbour of class *i* to feature vector *X* under test and $?_i$ the covariance matrix of class *i*. The smallest distance determines the class. This procedure is also referred to as the volumetric *k*NN procedure. For computational reasons (1) can be simplified to:

$$d_i^2 = \sum_{j=1}^{n} \frac{(x_j - x_{ij,NN})^2}{\sigma_{ij}^2}$$

Here $x_j$ is the $j$-th element of feature vector $X$ with dimension $n$. $x_{ij,NN}$ is the $j$-th element of the $k$-th nearest neighbour $X_{i,NN}$ and $\sigma_{ij}$ is the standard deviation of all j-th elements of all sample feature vectors of class $i$.

Another example of a non-parametric classifier is the knowledge-based or rule-based classifier ([11], [15]). The most common rule in such classification is the *if-then* rule: *if* condition *then* inference. Fuzzy rule-based systems follow the same rules, except that conditions are not hard. In a basic rule-based system for instance the condition if a radar tone is dark, is determined by a hard threshold on the radar backscatter. In a fuzzy rule-based system this condition is determined by a membership function which describes the degree of membership to a fuzzy set ([2]). Because land cover and land use have many-to-many relationships, knowledge-based or rule-based classification systems are ideal to convert one to another ([11], [4]).

The classification techniques that were discussed, can be applied to pixels or objects. From a land-cover point of view, an object is a region or segment in which the feature space is homogeneous to some degree, so it fits one (physical) description. From a land-use point of view an object fits one function. In general, object-based classification is preferred to pixel-based classification (i.e. pixel-by-pixel classification) because it is more accurate ([13]). Objects can be extracted from the satellite image using segmentation techniques ([14], [2]).

## Change Detection Techniques

Change detection is useful when we have to update a map of an area and do not know which parts have changed. Several techniques exist which can be divided into two categories: pre-classification and post-classification change detection. The basic pre-classification methods are image differencing and image ratioing ([17], [16]), which compare the images directly, before classification. Image differencing is subtracting the before image from the after image. Image ratioing is dividing the first by the latter. Generally, image ratioing is less sensitive to radiometric errors, and preferred in case of radar change detection due to the radar image statistics. Other pre-classification methods, based on image differencing and ratioing, have been designed ([17], [5]). One of them, especially designed for radar images, applies an adaptive filter to the ratio image, to reduce the speckle-noise which is typical for radar images.

Post-classification change detection is applied to the classification results of images. The advantage of this method is that it applies to information from sources that are difficult to combine before classification (e.g. optical/infrared, radar, digital maps). A disadvantage of this method is that it is sensitive to classification errors.

The change detection techniques discussed can be applied to pixels and objects as well. Matching the results with existing map objects will show which areas or objects have changed. In object-based change detection it is important that the boundaries of the object under test are the same. Otherwise sliver or larger polygons that indicate false change may occur. One way to overcome this problem is to apply multi-channel segmentation in which both images are input to the segmentation process ([3], [18]).

## Results and Discussion

To investigate the effects of these classification and change detection techniques, several were applied to the objects that were recognised from data that were discussed. Addressed were spectral and textural features. Non-parametric techniques were applied because most features, especially the texture measures, appeared to be non-Gaussian distributed. The classifiers were trained manually by selecting a number of appropriate training objects. Table 2 gives an overview of the results.

Land use classification of Landsat 5 TM was done using a fuzzy rule-base which included rules based on the spectral intensities, texture (i.e. standard deviation) and some rules considering the relation between adjacent objects. Figures 2 and 3 shows the result. The relatively high percentage of correct classification (Pcc) is mainly due to the high number of bands compared to the low number of classes. Besides, this area is less complex because it is dominated by natural land cover instead of urban. The effect of using a fuzzy classifier must not be overestimated. The rules on the spectral intensities and texture are based on training sample-areas, as with the Nearest Neighbour (NN) classifier.

| sensor | nr.of bands | res. | map | scale | classifier | texture | nr.of class. | Pcc | kappa |
|---|---|---|---|---|---|---|---|---|---|
| Landsat 5TM | 6 | 30m | vMapl | 250K | fuzzy | yes | 4 | 82.9 | 59.6 |
| ERS 1 | 1 | 30m | VMap1 | 250K | NN | yes | 5 | 52.1 | 36.3 |
| Ikonos | 4 | 4m | TOP10 | 10K | NN | no | 5 | 42.4 | 26.8 |
| PHARUS | 1 | 4m | TOP10 | 10K | NN | yes | 3 | 48.3 | 29.4 |

Table 2: Overview of land-use classification experiments.

The ERS 1 image was classified using different texture measures because only one band was available. The best performing texture measures (i.e. mean intensity, variance, weighted-rank fill ratio and semi-variograms) were applied. The results show that the textures improve classification but the results are not optimal. This is due to the fact that (i) the class definitions between the map (land use) and the image (land cover) are not exactly the same, (ii) the fact that the map shows deficiencies, and (iii) the fact that the land-cover information content of ERS 1 leaves something to be desired.

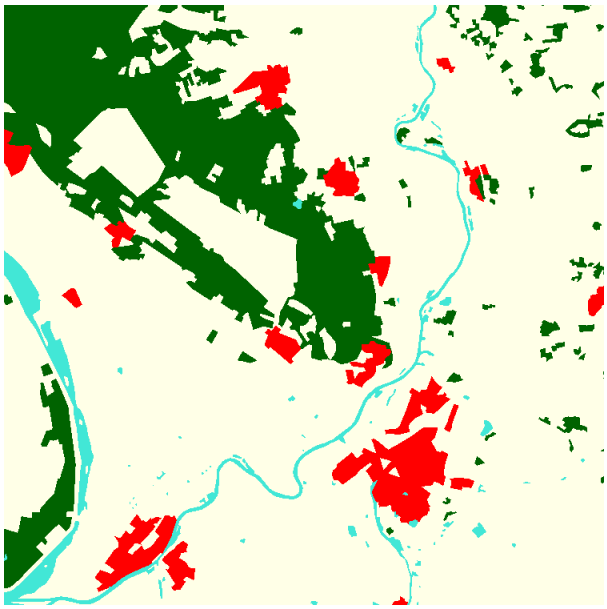


Figure 2: Classification of the test area Zwolle and Veluwe. Urban = red; forest = green; water = blue; other = light yellow; unclassified = black.

The classification result of Ikonos shows the lowest Pcc. The reason for this was found in the facts that different land covers are made up of the same materials (e.g. roofs and roads are often made up of tarmac) and that the map that was used showed deficiencies as well. Using texture did not improve the result. The result is worse than that of Landsat 5 TM which is due to the lower number of bands, the higher complexity of the scene, and the higher number of classes. Again, it can not be fully ascribed to the classifier.

The classification accuracy of PHARUS, using the same map, was slightly better. The best performing texture measures were applied here as well. However, the result was not better than that of ERS 1. Besides the reasons mentioned with the classification of ERS 1, this was caused by the fact that radar reflections are often due to parts of buildings instead of the whole building (e.g. wall-ground reflections).

The NN classification of the high-resolution sensors Ikonos and PHARUS is not optimal. Classifying high-resolution imagery is clearly more difficult than classifying low-resolution images. On the other hand, different maps were used for different resolutions.

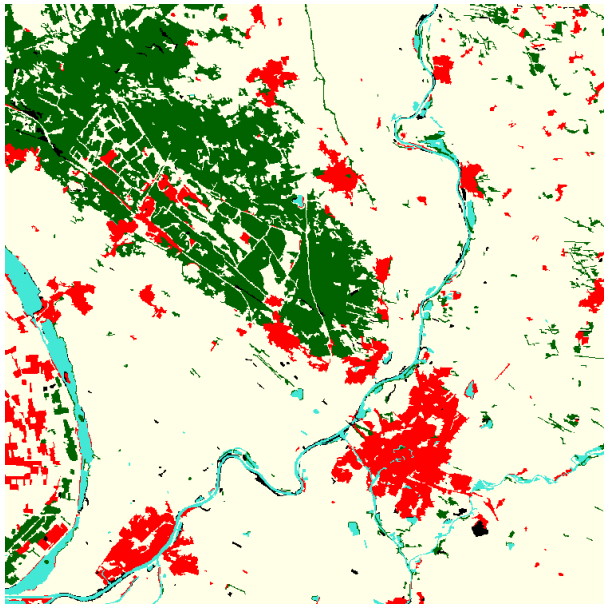


Figure 3: VMap level 1 of the test area Zwolle and Veluwe (source: Topografische Dienst Kadaster) Landsat 5 TM image.

In case of the change detection techniques, three were applied to the two PHARUS images:

- Pre-classification change detection applied to the ratio image using an adaptive filter
- Pre-classification change detection by multi-channel segmentation
- Post-classification change detection

Pre-classification techniques are preferred to post-classification change detection, unless the classification is accurate but this was not the case, see table 2. The difference between pre-classification change detection applied to the filtered ratio image and by multi-channel segmentation is small. The first slightly better preserves smaller objects, while the latter better reproduces the shape of the changed objects. The main reason for that is found in the fact that the speckle in the PHARUS images was already quite low due to its relatively high number of looks.

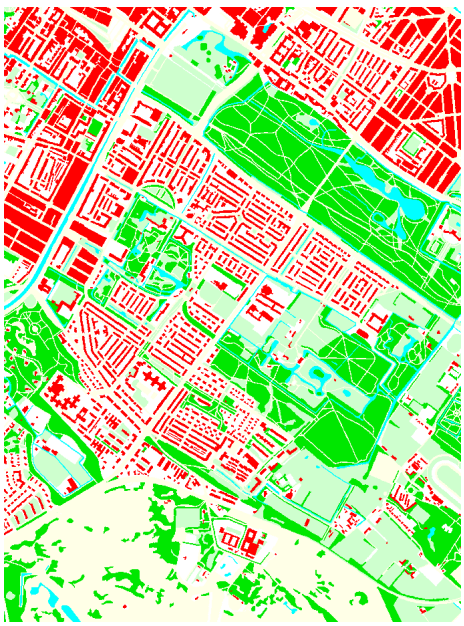


Figure 4: Classification of the generalised TOP10vector of The Hague. Buildings = red; roads/bare soil = light yellow; low vegetation = light green; trees = dark green; water = blue; other land use = white, shadow/unclassified = black.

## Conclusions

Despite the fact that it is not easy to make a good comparison between the sensors due to different numbers of bands, areas, maps, scales and numbers of classes, the results are indicative for what can be achieved in image classification and change detection today. Although the methods can be improved, more information is required in the map-updating process. Instead of using a single sensor, images from multiple sensors covering different parts of the spectrum can be applied (e.g. combine optical/infrared with radar). Another possibility is to include information on the relation between adjacent objects (this was only done for Landsat 5 TM) or information from the map that has to be updated. Techniques that improve the vector results of classification may be required as well.

Map updating may not become fully automatic, but the job of a human operator can be made easier using the techniques investigated in this paper. Change detection will reduce the number of areas an operator has to check for changes, and object-based classification of those areas will provide the operator with the new map objects. And although the objects may not all be perfectly shaped and classified, the operator does not have to do all the work, especially when the percentage of correct classification is high.

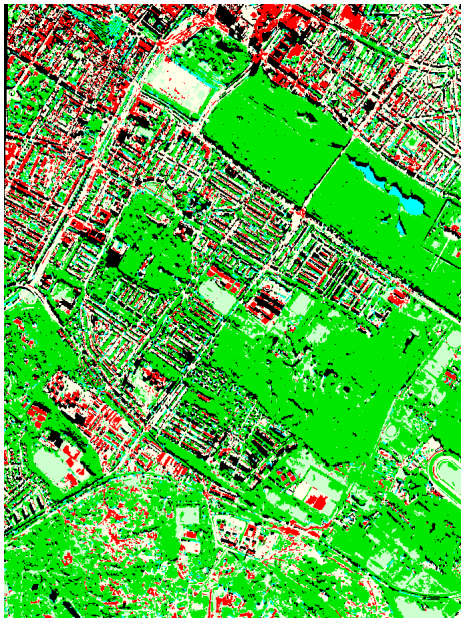


Figure 5: Generalised TOP10vector of The Hague (source: Topografische Dienst Kadaster) Ikonos image.

## References

[1] M.J. Barnsley, L. Moller-Jensen, S.L. Barr, *Inferring Urban Land Use through Spatial and Structural Pattern Recognition*, Remote Sens-

ing and Urban Analysis - GISDATA 9. **115-144**, 2001.

[2] U.C. Benz, P. Hofmann, G. Willhauck, I. Lingenfelder, M. Heynen *Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information*, ISPRS Journal of Photogrammetry & Remote Sensing. Vol. 58, **239-258**, 2004.

[3] R.G. Caves, S. Quegan, *Multi-channel SAR Segmentation: Algorithm and Applications*. Proc. European Symp. on Satellite Remote Sensing II, Paris, **241-251**, 1995.

[4] J. Cihlar, L.J.M. Jansen, *From Land Cover to Land Use: A Methodology for Efficient Land Use Mapping over large Areas*, Professional Geographer, Vol. 52, No. 2, **275-289**, 2001:

[5] R.J. Dekker, *Speckle filtering in satellite SAR change detection imagery*. International Journal of Remote Sensing, Vol. 19, No. 6, **1133-1146**, 1998.

[6] R.J. Dekker, *Object-based updating of land-use maps of urban areas using satellite remote sensing.* MSc thesis, Vrije Universiteit Amsterdam, The Netherlands, 2003a

[7] R.J. Dekker, *Texture Analysis and Classification of ERS SAR Images for Map Updating of Urban Areas in The Netherlands.* IEEE Transactions on Geoscience and Remote Sensing, Vol. 41, No. 9, **1950-1958**.

[8] DGIWG, 2000: DIGEST Edition 2.1. Digital Geographic Information Working Group (DGIWG). http://www.digest.org/ [accessed 21 March 2004]

[9] P.F. Fisher, A.J. Comber, R. Wadsworth, *Land use and Land cover: Contradiction or Complement*. In: Re-Presenting GIS. D. Unwin (editor), Wiley, Chichester, 2002

[10] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Second Edition. Academic Press, San Diego, CA, 1990

[11] Gong , P., and Howarth, P.J., *Land Cover to Land Use Conversion: a Knowledge-Based Approach. Annual Conference of ASPRS*, Denver, Colorado, Vol. 4, **447-456**, 1990

[12] H.S.F. Greidanus, P. Hoogeboom, P.J. Koomen, P. Snoeij, H. Pouwels, *First Results and Status of the PHARUS Phased Array Airborne SAR*. IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Lincoln, Nebraska, **1633-1635**, 1996

[13] L.L.F. Janssen, M.N. Jaarsma, E.T.M. van der Linden, *Integrating Topographic Data with Remote Sensing for Land-Cover Classification*. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 11, **1503-1506**, 1990

[14] C.J. Oliver, *Review article: information from SAR images*, J. Phys. D: Appl. Phys., Vol. 24, **1493-1514**, 1991

[15] J.A. Richards, *Remote Sensing Digital Image Analysis: An Introduction*, Second Edition. Springer-Verlag, Germany, 1993

[16] E.J.M. Rignot, , J.J. van Zyl, *Change detection techniques for ERS-1 SAR data*. IEEE Transactions on Geoscience and Remote Sensing, Vol. 31, No. 4, **896-906**, 1993.

[17] A. Singh, Digital change detection techniques using remotely-sensed data, review article. International Journal of Remote Sensing, Vol. 10, No. 6, **989-1003**, 1989

[18] G. Willhauck, *Comparison of object oriented classification techniques and standard image analysis for the use of change detection between SPOT multispectral satellite images and aerial photos*, International Archives of Photogrammetry and Remote Sensing, Amsterdam, The Netherlands, Vol. XXXIII, Supplement B3, **35-42**, 2000.

## Contact Information

R.J. Dekker
TNO Defence, Security and Safety
P.O. Box 96864
2509 JG The Hague
The Netherlands
Phone +31 70 374 04 31
Fax +31 70 374 06 54
rob.dekker@tno.nl

# Analytical Software Design Case MagLev Stage Software Project for Philips Applied Technologies

Guy H. Broadfoot, George Kielty

*Product innovation, quality and time to market are key elements in the battle to achieve and sustain competitive advantage. For a growing number of businesses, this means software development. Software is now an essential component embedded in an ever increasing array of products. It has become an important means of realising product innovation and is a key determinant of both product quality and time-to-market. For many businesses, software has become* business-critical *and software development is a* strategic *business activity. At the same time, software development continues to suffer from poor predictability. Existing development methods appear to have reached a quality ceiling that incremental improvements in process and technology are unlikely to breach. To break through this ceiling, a different approach is needed. In this paper, we describe how Verum applied Analytical Software Design (ASD), a new approach that applies software engineering mathematics to industrial software development, to develop the software controlling an advanced mechatronics subsystem being developed by Philips Applied Technologies in the Netherlands.*

"*ASD is a formal method that is informal enough to be applied in practice.*" G.P.M. Haagh
Senior Software Architect, Philips Applied Technologies

## Introduction

Product innovation, quality and time to market are key elements in the battle to achieve and sustain competitive advantage. For a growing number of businesses, this means software development. Software is now an essential component embedded in an ever increasing array of products. It has become an important means of realising product innovation and is a key determinant of both product quality and time-to-market. For many businesses, software has become *business critical* and software development is a *strategic* business activity.

Today, software development continues to suffer from poor predictability. Business managers need reliable answers to the questions "When will it be ready?", "What will it cost?" and "How well will it work?" These are the very questions that software developers are least able to answer.

In recent years, in an attempt to overcome these problems, companies have invested heavily in software development process improvements, technology, infrastructure and training. In spite of this, the rapidly increasing complexity and amount of software still presents a serious challenge. According to studies, 40% - 50% of total development costs are typically lost on avoidable rework [4]; 15% - 25% of software defects are delivered to customers [4]; in 2002, software failures cost the U.S. economy an estimated $59.5 billion [7].

Existing development methods appear to have reached a quality ceiling that incremental improvements in process and technology are unlikely to breach. The amount and complexity of in-product, embedded software is growing exponentially; according to the SEI, productivity in the most successful organisations is improving linearly, at best. The challenges many businesses experience developing embedded software and being able to guarantee its quality and correct functioning is a testament to this growing capability gap and inability of current testing-centric software development practices to bridge it. To bridge this divide, a different, more formal, approach is needed.

## The Case: The MagLev Stage

Philips Applied Technologies[1] is part of the Dutch Royal Philips Electronics group of companies. Part of its mission is to develop innovative solutions for advanced manufacturing. It is a leading developer of industrial vision systems and high precision mechatronic systems.

One of its latest products is a highly accurate, high performance "stage" known as the MagLev Stage (see figure 1). This is a subsystem designed to be incorporated in a variety of industrial systems that require high speed, highly accurate material positioning, especially in high vacuum situations, for applications such as e-beam inspection and laser cutting. It uses advanced magnetic levitation servos and achieves a repeatable sub-micron scanning accuracy (130 Nanometers or better).

An essential part of the MagLev Stage is the control software embedded in it. This software coordinates the actions of the multi-axis controllers and provides an Application Program Interface (API) to customer developed domain specific application software.

Early in 2004, a "proof of concept" version of the control software was developed to enable the mechanical and electronic sub-systems to be developed and tested. This development took about 10 weeks in order to achieve a level of "good weather" functionality useful for the product development. Although suitable for its purpose as a "proof of concept" prototype, this software was not of the industrial quality levels considered suitable for the final product. Since its initial development, defects have emerged at regular intervals, including software crashes and race conditions. By January 2005, more than 20 versions were present in the software configuration management system, each representing a major release to fix multiple errors.

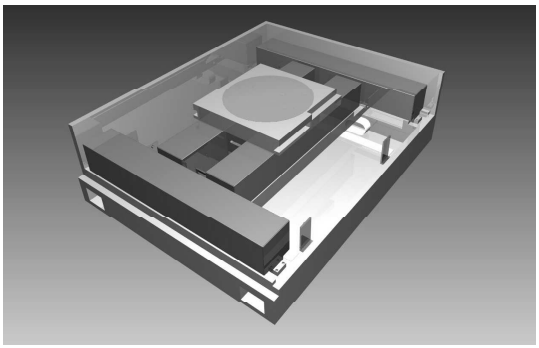


Figure 1: MagLev Stage

In September 2004, it was decided that a new version of the controller software would have to be developed in order to achieve required quality standards. Given the complex nature of the software, Philips Applied Technologies and Verum together applied Analytical Software Design (ASD) techniques in which the complete software design is modelled mathmatically and model-checked for correctness before implementation starts. After verifying the design mathematically, Verum's ASD techniques enabled 90% of the new code to be generated in C++ automatically from the verified design specifications.

## Technical Details

The MagLev Stage consists of two *substages* called the *Intermediate Substage* and the *Carrier Substage* respectively. Each substage is controlled by its own dedicated multi-axis controller. The multi-axis controllers are existing subsystems previously developed by Philips Applied Technologies and used in other products.

[1] `www.apptech.philips.com`

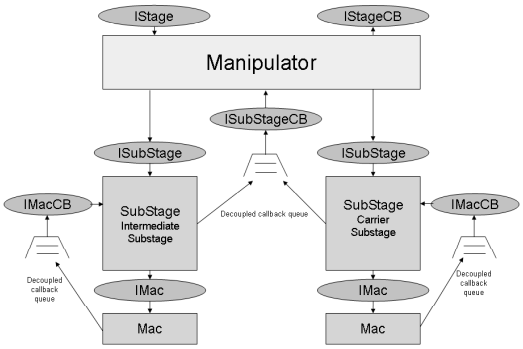



Figure 2: MagLev Software Overview

Figure 2 shows the overall organisation of the software into two major components, namely the *Manipulator* and the *SubStage*. In the diagram, software components are depicted by rectangles; major interfaces are depicted by the labelled ovals.

The SubStage component is responsible for controlling a single substage via its dedicated multi-axis controller known as a *MAC*. The diagram shows two instances of the SubStage component, one controlling the Intermediate Substage and one controlling the Carrier Substage. The Manipulator component coordinates and controls the two substage components. All actions that are specific to a single MAC are implemented in the SubStage component; all actions requiring coordination between the substages, such as most movements and all exception and error handling, are implemented in the Manipulator component. As is usual in such systems, the "good weather" behaviour, although complex, is relatively straight forward; the majority of the program logic is required to handle all the various exception conditions that can occur.

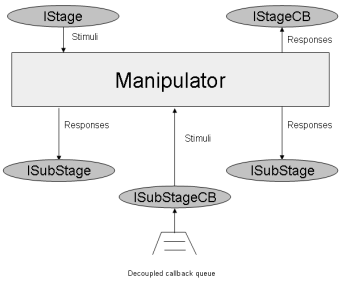



Figure 3: Manipulator Context Diagram

The Manipulator component implements the API to be used by the customer-developed domain specific application software. It must be "thread safe"; that is, able to support multi-threaded client applications while handling asynchronous call-back events from the two substages. For efficiency reasons, the Philips Applied Technologies senior architect required the execution architecture to minimise context switching with execution remaining under the caller's thread context as long as possible. Figure 3 is a context diagram of the Manipulator component. This shows the Manipulator as implementing its client API (IStage), sending asynchronous call-back notifications to the client application (IStageCB), using the SubStage API (ISubStage) and receiving notifications from the two substages via the ISubStageCB call-back interfaces. All the ISubStageCB events are routed to the Manipulator via a queue and are executed under the context of a separate deferred procedure call (DPC) server thread.

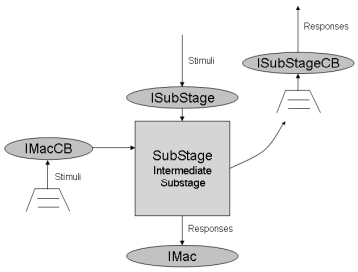



Figure 4: SubStage Context Diagram

Figure 4 shows the context of each SubStage component instance. It implements an API used by the Manipulator (ISubStage and ISubStageCB) and uses the interfaces provided by the MAC (IMac and IMacCB). The SubStage receives API calls from the Manipulator and asynchronous event notifications from the MAC via a queue and a separate DPC server thread. The ISubStage interface realises a high level abstraction of the physical substage, with high level moves implemented in terms of the primitive move operations provided at the MAC interface.

## An Overview of ASD

Analytical Software design is based on two design principles:

- Business critical software must be based on designs that are verified before implementation starts;
- 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 to 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 coverage or 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, mathematics are not routinely applied to 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 many kinds of errors, such as race conditions and deadlocks, this is also the least certain way to find them.

Analytical Software Design[2] 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 uses the Sequence-based Specification method [8, 9] 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 rigour and precision necessary for mathematical analysis.

ASD applies the Box Structured Development Method [5, 6] following the principles of stepwise refinement to transform the black box design specifications into state box specifications from which programming is based.

The ASD Model Generator generates mathematical models from the black box and state box specifications and designs automatically. These models are generated in the process algebra CSP [3, 10] and can be formally analysed and verified using the model checker FDR [1]. For example, we can use the model checker to verify (i) whether a design satisfies its functional requirements; (ii) whether the state box coding specification is behaviourally equivalent to the black box design; and (iii) whether the design uses other components according to their external functional specifications.

The ASD Code Generator can generate significant amounts of 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. Code generation may not be applicable to every project but in those cases where it is, significant development efficiency gains can be achieved.

ASD uses Statistical Testing methods based on Usage Models derived directly from the ASD Specifications to test software components against the verified designs. The ASD Test Case Generator and Analyser generates large numbers of self-running test cases and analyses their results.

Figure 5 shows the main elements of ASD. The functional specification is analysed using the Sequence-based Specification method extended to enable nondterminism to be captured. This enables the externally visible behaviour of the system to be specified with precision and guarantees completeness.

[2] Patent applied for under patent application number GB 0410047.5

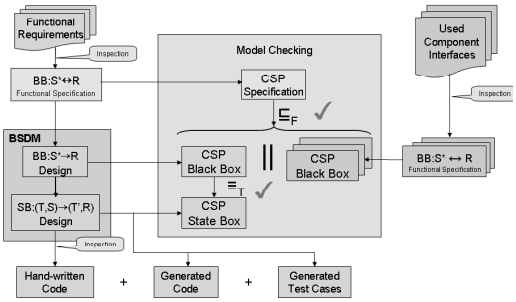



Figure 5: ASD Overview

Because ASD specifications avoid difficult mathematical notations and are fully traceable to the original specifications, they can be validated by inspection with project stakeholders. Next, the design is specified using Sequence-based Specification. This still remains a creative, inventive design activity requiring skill and experience combined with domain knowledge. With ASD, however, the design is typically captured with much more precision than is usual with conventional development methods, raising many issues early in the life cycle and resolving them before implementation has started.

The ASD model Generator is used to generate process algebra models of both the specification and the design so that the design can be verified for compliance with the specification. In most cases, a design cannot be verified in isolation; it depends on its execution environment and the components it uses for its complete behaviour. In ASD, used component interfaces are specified using Sequence-based Specification, the corresponding mathematical models are generated using the ASD Model Generator and these models, combined with those of the design, are verified for compliance with the specification. For CSP models, this verification is done mathematically using the model checker FDR. Errors detected during the verification are corrected in the design specification, new CSP models are generated and the verification is repeated. (This is typically a very rapid cycle.)

When the design has been verified, the ASD Code Generator 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 suggests this is typically between 70% and 90%, but it can be lower on some projects.

Finally, from the same set of design specifications, large numbers of statisitically selected test case can be generated in the form of self running tests and the results analysed by the ASD Test Analyser.

## Applying ASD to the MagLev Stage Development

The design team consisted of a senior software architect and software engineer from Philips Applied Technologies and two employees from Verum. The goals of the project were:

1. To re-develop the MagLev Stage control software to industrial quality standards as quickly as possible;
2. For Philips Applied Technologies to gain practical experience of applying ASD in practice with a view to assessing its applicability to other typical software developments carried out within Philips Applied Technology.

The work proceded as follows: firstly, an ASD specification of the MAC interface (IMac and IMacCB) was made based on existing specifications and the expert knowledge of the senior architect. This black box function was plotted in the form of a state transition diagram and reviewed by the team.

Next, an ASD specification of the client application API (IStage and IStageCB) was made based on the existing implementation and with frequent references to the existing code. The process of making the ASD specification raised a significant number of specification issues, most of which were resolved by the senior architect based on his extensive domain knowledge and experience gained in developing the original “proof of concept” prototype.

The architecture was then developed, partitioning the major functions of the control software between the Manipulator component and the two instances of the SubStage component; an ASD specification of the SubStage interfaces (ISubStage, ISubStageCB) was made, reflecting the first “guess” at the SubStage abstraction.

The first major design task was the design of the Manipulator. This was specified using Sequence-based Specification. As the design evolved, the precision of the ASD interface specifications of

the client interface and the SubStage interface was extremely beneficial. As the design neared completion, the ASD Model Generator was used to generate the CSP models of the client interfaces (IStage, IStageCB), the SubStage interfaces (ISubStage, ISubStageCB) and the Manipulator design. The parallel composition of the Manipulator design model plus two instances of the SubStage Interfaces (one for the Intermediate SubStage and one for the Carrier SubStage) were verified against the client interface model using the model checker. This was done after first verifying with the model checker that the design was free from divergence, internal inconsistencies and deadlocks.

During this process, many design and some specification errors were discovered by the model checker. As they were discovered, the appropriate ASD specification was updated to correct the error, new mathematical models generated and the verification continued. This cycle occurs quite rapidly, finding and fixing several errors per hour. This differs significantly form conventional, testing-based development method. Unlike conventional testing:

- All of this is done without having written any program code or executing any test cases.
- This form of verification is based on mathematical proof and is total. It is equivalent to 100% *execution path* coverage, something unacheivable by testing.
- This form of verification is particularly good at uncovering dynamic behavioural errors such as deadlocks, race conditions and design behaviour that violates interfaces specifications. Such errors are extremely difficult to detect and diagnose using conventional testing because their nondterministic nature makes them to reproduce and repair.
- This verification is done before investing in implementation, at the most economic point in the life-cycle.

When the Manipulator design was completed and verified, work began on the SubStage design. Again, this was specified using the Sequence-based Specification Method. In this case, the implemented interface is the SubStage interface (ISubStage, ISubStageCB) and the used interface is the MAC interface (IMac, IMacCB). As the design neared completion, the CSP models were generated automatically using the ASD Model Generator and FDR was used to check the SubStage design plus the MAC interface against the SubStage interface. The SubStage interface model was the same one used to verify the Manipulator design.

During the SubStage design, it proved impossible to implement the SubStage interface exactly as it had been specified and it was necessary to change it. This involved changing the ASD SubStage interface specification, regenerating its mathematical model and then verifying the Manipulator design against the changed SubStage interface to assess the impact of the changes on the Manipulator design. Where necessary, the Manipulator design was changed and verified against the modified SubStage interface specification. This enabled the impact of ISubStage design alternatives on the Manipulator to be assessed quickly and provided additional input for making often difficult technical choices.

When both the Manipulator and SubStage designs were completed and verified by model checking, the C++ code of both the Manipulator and th SubStage was generated using the ASD Code Generator.

## Results

The ASD specification of the MAC interfaces took about 1 week; the specification of the client API interface and the SubStage interface took about the same time. The MAC interface specification has 493 transition rules and 12 canonical sequences, the longest of which is 5 stimuli long. The ASD specification of the client API has 345 transition rules and 13 canonical sequences. The SubStage interface has 370 transition rules and 13 canonical sequences.

The ASD design and verification of the Manipulator took about 4 weeks to complete. The design was extremely complex due to the complex behaviour of the MAC as this was still visible at the SubStage interface plus the event driven and concurrent nature of the behaviour. Due to its complexity, the Manipulator design was hierachically decomposed into a top level design together with 3 significant lower level sub-designs. In total, the design has 1,700 transition rules and 28 canonical sequences. This hierarchical design structure was carried through into the generated mathematical models and the generated C++ code, providing full traceability between these different views.

During the design verification of the manipulator, about 200 errors were detected in the model checking phase. Most of these fell into one of two categories: i) internal inconsistencies where the design violated the interface specifications of the used components or was unable to react correctly to notifications arriving asynchronously from the used interfaces; ii) race conditions that were particularly difficult due to a) the number of unstable states in the SubStage specification arising from the nature of the underlying MAC behaviour and b) the loose coupling between the Manipulator and the SubStage introduced by the event notification queue mechanism. The order of verification was as follows: i) to verify freedom from race conditions, divergence and deadlocks; ii) to verify complicance with the used SubStage interface specifications; iii) to verify compliance with the client API specifications, the interface implemented by the design.

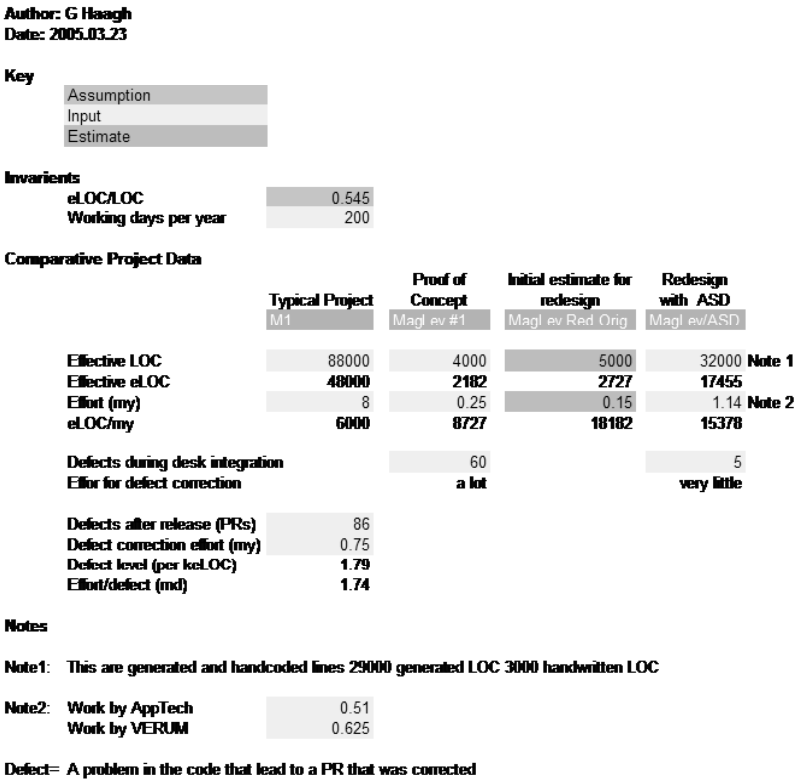
The ASD design and verification of the SubStage took about 4 weeks to complete. Due to its complexity, the design was hierarchically decomposed into a top level design plus 5 lower level subdesigns. In total, the design has 4,700 transition rules and a total of 84 canonical sequences. The order of verification was the same as described above. Again, the number of unstable states in the behaviour of the MAC together with the decoupling caused by the event notification queue resulted in a very complex design with many possibilities for race conditions and other unexpected behaviour. During the verification, in the order of 200 errors were detected by model checking and removed.

After all designs were completed and mathematically verified, the C++ code was generated. The generated code is structured according to the well known State Pattern [2] and was tailored to meet the code architecture required by Philips Applied Technologies. This is a normal part of the ASD code generation process; experience shows that "standard" code generators are frequently too inflexible in the style and strucure of the code they generate. Every project and development environment has specific requirements for the generated code to ensure that it properly integrates with the rest of the code base and the run-time platform. In this project, the run-time platform was VxWorks. In total 17,000 executable lines of code were generated, representing more than 90% of the code. The hand written code was either concerned with domain specific issues such as coordinate transformations or "glue" code interfacing the software to the rest of the run-time environment. Although the final run-time platform was VxWorks, component testing was done by Verum under Windows XP. Testing in the final VxWorks environment is being performed by Philips Applied Technologies.

The comparative results of this project are shown in figure 6. Philips Applied Technologies calculates that its code production rate for a typical software development, including design, specification, coding and testing effort, is about 6,000 executable lines of code per man year. The original MagLev "proof of concept" software was produced at a rate of 8,727 executable lines of code per man year. Desktop integration testing of this software, using simulated hardware, found 60 defects, resulting in a large - but undocumented - amount of rework.

In 2004, when considering the redesign of the MagLev software using traditional methods, the Applied Technologies design team expected to produce approximately 5000 lines of code in 6 man weeks: a productivity equivalent to 18,000 executable lines of code per man year. Based on their experience with the "proof of concept" version, they also obviously expected an increase in the quality of the end result.

Ultimately the redesign of the MagLev software was performed together with employees of Verum using ASD. The result was the production of 17,000 executable lines of code in 45 man weeks of effort, including all specification, design, design verification, coding and desktop integration testing effort. This equates to a production rate of 15,000 executable lines of code per man year. The stated effort captures the contributions from both Applied Technologies design staff and Verum's employees. It also captures the learning curve needed by both parties; Verum's employees to learn about the MagLev application and Applied Technologies engineers to learn how to work with ASD. Much of this learning curve would not be required for future projects. Furthermore, application of ASD to the design of this system resulted in the discovery of about 400 defects during design verification. The average effort to find and fix each defect was approximately 1 man hour per defect. Consequentially the software delivered to Philips Applied Technologies has a very low defect rate. During desktop integration



**Author: G Haagh**
**Date: 2005.03.23**

**Key**

| |
|---|
| Assumption |
| Input |
| Estimate |

**Invariants**

| | |
|---|---|
| **eLOC/LOC** | 0.545 |
| **Working days per year** | 200 |

**Comparative Project Data**

| | Typical Project M1 | Proof of Concept MagLev #1 | Initial estimate for redesign MagLev Red Orig | Redesign with ASD MagLev/ASD | |
|---|---|---|---|---|---|
| **Effective LOC** | 88000 | 4000 | 5000 | 32000 | **Note 1** |
| **Effective eLOC** | **48000** | **2182** | **2727** | **17455** | |
| **Effort (my)** | 8 | 0.25 | 0.15 | 1.14 | **Note 2** |
| **eLOC/my** | **6000** | **8727** | **18182** | **15378** | |
| **Defects during desk integration** | | 60 | | 5 | |
| **Effor for defect correction** | | **a lot** | | **very little** | |
| **Defects after release (PRs)** | 86 | | | | |
| **Defect correction effort (my)** | 0.75 | | | | |
| **Defect level (per keLOC)** | **1.79** | | | | |
| **Effort/defect (md)** | **1.74** | | | | |

**Notes**

**Note1:** **This are generated and handcoded lines 29000 generated LOC 3000 handwritten LOC**

| | | |
|---|---|---|
| **Note2:** | **Work by AppTech** | 0.51 |
| | **Work by VERUM** | 0.625 |

**Defect= A problem in the code that lead to a PR that was corrected**

Figure 6: Comparitive Results

testing with simulated hardware, only 5 errors were found and very little effort was required to correct these errors.

Of course, the number of executable lines of code is a poor indicator of the complexity of a piece of software. Comparison of hand versus automatic code generation techniques leads to a discussion of the relative efficiency of each technique, with no obvious conclusions except that automatically generated code leads to far lower error rates. Unfortunately, there are no other common metrics that give an indication of complexity in this case. However, the design team judged the complexity of the MagLev design to be at least twice that estimated at the beginning of the project, even with the experience of having produced a proof of concept version.

As a result, Philips Applied Technologies drew the following conclusions from the application of ASD to the redesign of the MagLev software:

- Overall the use of ASD in the design/code/unit test phases is cost neutral w.r.t. traditional ways of working; that is, the benefits were gained at no extra cost as compared to traditional working methods.
- The number of defects found during desktop integration is reduced by a factor 12
- The perceived quality of the code is MUCH higher (supported by the figures)

Verum's employees also observed that:

- The complexity of the MagLev (re)design problem was much greater than that anticipated by the Applied Technologies design team
- The use of ASD exposed the complexity of the (re)design problem during the earliest moments of the development
- The MagLev software was delivered on time according to original expectations
- The MagLev software was delivered in line with effort estimates, bearing in mind the unexpected complexity of the design problem.

At the time of writing this report, the MagLev software remained to be tested with real hardware and released to real customers. Therefore Applied Tech-

nologies has only measured the effect of ASD on early lifecycle phases and has yet to experience the benefits ASD brings to system testing, release and maintenance.

## Conclusions

This project has demonstrated to Philips Applied Technologies:

1. The application of ASD is cost neutral over conventional design methods during the first half of the project lifecycle.
2. The application of ASD results in a factor 12 reduction in defects found during initial integration testing.
3. The application of ASD results in a predictable completion date for the project.
4. ASD specifications are understandable and usable by project stakeholders without knowledge of software engineering mathematics; there is no complex mathematical notation to be learned.
5. ASD enables experienced employees of Verum to work productively together with domain experts in a joint design team in an existing software development environment and with software engineers and architects not specifically trained in the method.
6. ASD is applicable to a wide variety of projects within Philips Applied Technologies.
7. ASD results in designs and implementation of a much higher quality than can be achieved by conventional methods.

The senior architect on this project stated that he has a much higher level of confidence in the quality than he has using conventional methods. He said: *"This is the first formal method informal enough to be applied in practice."*

## Acknowledgements

We are grateful to Philips Applied Technologies[3] for allowing us to present the case study and for their cooperation and support when we applied these techniques together to develop control software for the MagLev Stage. We are particularly indebted to G.P.M. Haagh Senior Software Architect and Rutger van Beusekom Software Engineer, both of Philips Applied Technologies, for their cooperation and positive contribution in applying ASD to this development.

## References

[1] Formal Systems (Europe) Ltd. *Failures-Divergence Refinement: FDR2 User Manual*, 2003. See `http://www.fsel.com`.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[3] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[4] Thomas McGibbon. A business case for software process improvement revised. Technical report, Data & Analysis Center for Software, 1999.

[5] H. D. Mills. Stepwise refinement and verification in box structured systems. *Computer*, 21(6):23–26, 1988.

[6] H. D. Mills, R. C. Linger, and A. R. Hevner. *Principles of Information Systems Analysis and Design*. Academic Press, 1986.

[7] The economic impacts of inadequate infrastructure for software testing. Technical report, National Institute of Standards and Technology NIST, US Department of Commerce, 2002.

[8] S. J. Prowell and J. H. Poore. Sequence-based software specification of deterministic systems. *Software - Practice and Experience*, 23(3):329–344, 1998.

[9] S. J. Prowell and J. H. Poore. Foundations of sequence-based software specification. *IEEE Transactions of Software Engineering*, 29(5):417–429, 2003.

[10] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.

[3] Philips Applied Technologies B.V., Eindhoven, The Netherlands.

## Contact Information

**Guy H. Broadfoot**
guy.broadfoot@verum.com

**George Kielty**
george.kielty@verum.com

**Verum Consultants B.V**
Paradijslaan 28-28a
5611 KN, Eindhoven
The Netherlands
Phone +31-40-2359090
Fax +31-40-2359099

# Recent OOTI publications

The post-masters program OOTI is concluded with a design project. The final reports of these projects are publicly available in general unless stated otherwise. The following reports are the latest publications.

Acuña, C.
*Volume and 3D graphics rendering in a curved MPR application*
ISBN: 90-444-0413-X

Best, C. van, and Peter, C.
*PDF filter framework*
ISBN: 90-444-0441-5

Cuppen, R.P.G.
*Onyx Graphics Security Overhaul: The Olympus System*
ISBN: 90-444-0466-0

Ilchenko, V.
*Design and implementation of a structured reporting system*
ISBN: 90-444-0412-0

Leeuwen, J.J.A. van
*Task allocation in a mixed hardware-software environment*
ISBN: 90-444-0428-8

Liu, X. an Srinivasan, S.
*MG-R identification of component boundaries*
ISBN: 90-444-0419-9

Lukin, S.
*Analysis tool support for WIS engineering in Hera project*
ISBN: 90-444-0418-0

Michael, G.
*Analyzing run-time component memory consumption with aspect-oriented techniques*
ISBN: 90-444-0361-3

Mulyar, N. and Posta, L.
*Coupling of multidisciplinary models*
ISBN: 90-444-0417-2

Mumford, E.
*User adaptive video transmission man-agement over a wireless network*
ISBN: 90-444-0417-2

Nicolaas, A.R.
*Tagnology in the CE context*
ISBN: 90-444-0414-8

Paštrnak, J.
*Advanced efficient interrupt handler and JPEG decoding framework for DXC platform*
ISBN: 90-444-0422-9

Peters, R.G.O.
*Radio on Unicorn*
ISBN: 90-444-0442-3

Qi, Y.
*Interface Recorder and Player Project*
ISBN: 90-444-0348-6

Siahaan, D.O.
*The Cassandra project: metadata data-base communication infrastructure and its implementation*
ISBN: 90-444-0435-0

Tjiong, M.
*Agent-based architectural pattern*
ISBN: 90-444-0431-8

Yakimovich, A.
*Asynchronous (handshake technology) on-chip interconnect infrastructure*
ISBN: 90-444-0427-X