

VU Research Portal

Bit-to-board analysis for IT decision making

Kwiatkowski, L.M.

2012

document version

Publisher's PDF, also known as Version of record

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Kwiatkowski, L. M. (2012). *Bit-to-board analysis for IT decision making*.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:

vuresearchportal.ub@vu.nl

Bit-to-board analysis for IT decision making presents research concentrated on the information retrieval from bit-level data, like source code, and the use of the information in supporting IT decision making in the context of large organizations.

It demonstrates how information useful in, for instance, reduction of costs of running software applications can be collected at low cost from the already existing data.

Real-world software portfolios comprising millions of lines of code and supporting a large international corporation were used to illustrate that mining large quantities of business data can be done without much effort.



Łukasz Kwiatkowski

Bit-to-board analysis for IT decision making

Bit-to-board analysis for IT decision making

```
int find(int* a, int low, int high, int k) {
    while (low <= high) {
        int mid = (low + high) / 2;
        if (k < a[mid])
            high = mid - 1;
        else if (k > a[mid])
            low = mid + 1;
        else
            return mid;
    }
    return -1;
}

void insertSort(int* a, int size){
    for(int i=1; i<=size; i++){
        int j=i-1;
        while(j>=0 && a[j]>a[j+1]){
            int tmp=a[j];
            a[j]=a[j+1];
            a[j+1]=tmp;
            j--;
        }
    }
}

int main() {
    int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int k = 5;
    int low = 0, high = 10;
    int result = find(a, low, high, k);
    if (result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found\n");
    return 0;
}
```

Łukasz Kwiatkowski