



Software Estimation

Naba Kr Das



"You can't control what you
can't measure." *[Demarco 82]*

Why Estimation?

| Project | Estimates | | Schedule | | Status at Completion |
|------------------------------|-----------|------|----------|------|---------------------------|
| | Initial | Last | Initial | Last | |
| PROM (Royalty Collection) | 12 | 21 | 22 | 46 | Cancelled after 28 Months |
| London Ambulance | 1.5 | 6 | 7 | 17 | Cancelled after 17 Months |
| London Stock Exchange | 60-75 | 150 | 19 | 70 | Cancelled after 36 Months |
| Confirm (Travel Reservation) | 56 | 160 | 45 | 60 | Cancelled after 48 Months |
| Master Net (Banking) | 22 | 80 | 9 | 48 | Cancelled after 48 Months |




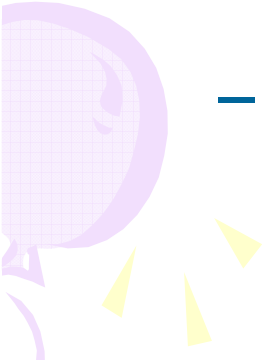
What was the Problem?



Under Estimation!



Under Estimation also lead to


- 
- Under Staffing
 - Under Scoping the QA Effort
 - Setting short Schedule
 - Staff Burnout
 - Low Quality
 - Loss of Credibility as Deadlines Missed
- 



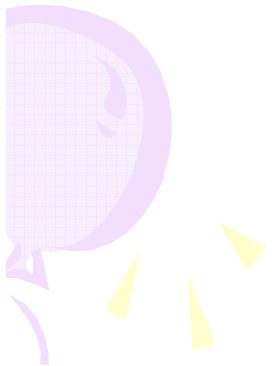
Over Estimation

- **Parkinson's Law**

"work expands so as to fill the time available for its completion."

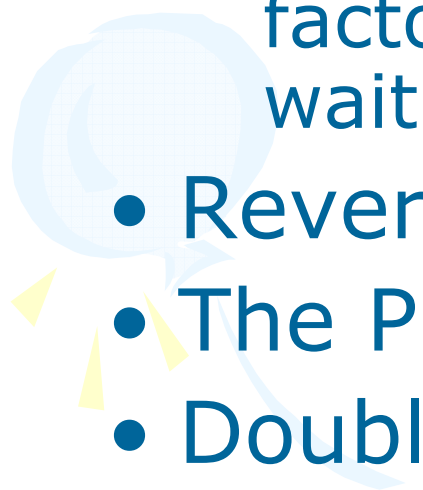



which means that the project will take as long as estimated even if the project was overestimated



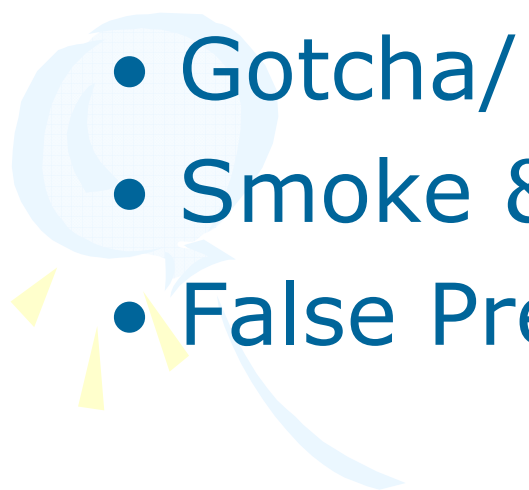
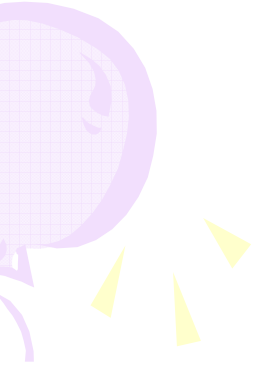


Game Theory in Estimation

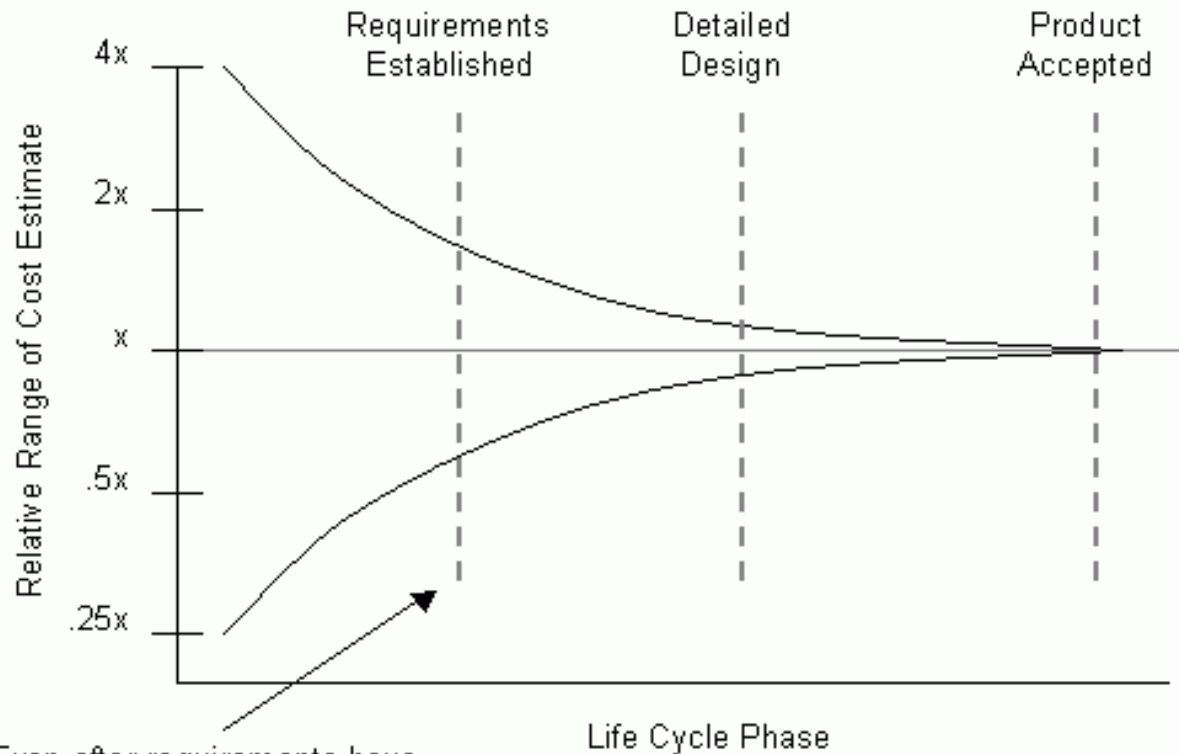
- Doubling & Add Some
 - Double the number and add fudge factor [average time lost in meeting, wait, talking and so on is 50%]
 - Reverse Doubling Option
 - The Price is Right/ Guess the Number
 - Double Dummy Spit
 - The X Plus Game
 - Spanish Inquisition
- 
- 



Game Theory in Estimation

- Low Bid/ What they are prepared to pay
 - Gotcha/ Playing the Pokies
 - Smoke & Mirror/Blinding with Science
 - False Precision
- 
- 

Cone of Uncertainty



Even after requirements have been established, estimates of total cost may be off by as much as 50%.

Precision increases throughout the software life cycle. [Boehm 1981]



Delphi Technique

- Consensus based forecasting technique developed at Rand Corporation in 1940.
- Based on Hegelian dialectic process
 - Thesis (establishing an opinion/view)
 - Antithesis (conflicting opinion or view) and
 - Synthesis (a new agreement or consensus)





Delphi Technique

- Goal is continual evolution towards 'oneness of mind`.
- Process consists of
 - Select a Facilitator and Panel of Experts
 - Maintain (preferably) Anonymity of the participants
 - Build a list of criteria based on Experts Opinion
 - Identify project constraints and preferences
 - Rank the Criteria and Calculate mean and deviation
 - Circulate to Expert and Re-rank the criteria
 - Analyze the result and Re-rank until stabilize




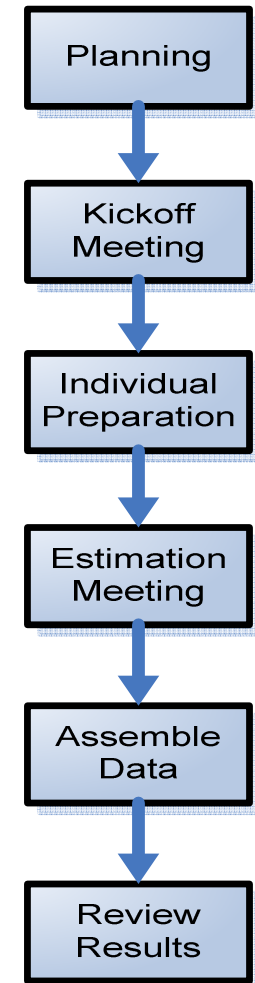
Wideband Delphi

- Consensus based estimation technique for estimating effort
 - Developed by Boehm et al in 1970 – variant of Delphi
 - Help build a complete task list/ WBS
 - The consensus approach helps eliminate bias in estimates
 - Can be used to estimate virtually anything
- 
- 



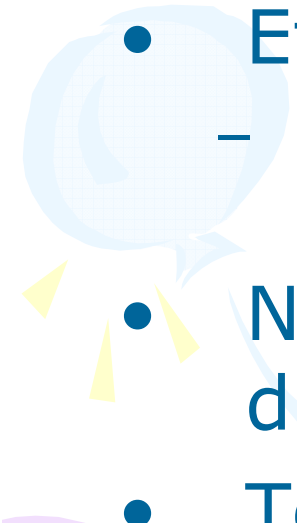

Wideband Delphi

- Choose the team – 3 to 7 experts
 - Kickoff meeting – generate WBS and circulate estimation form
 - Individual preparation – member generate individual estimation
 - Estimation Meeting – Iterative steps to gain consensus, use white board
 - Assemble Data – Compile the final tasks list, estimates & assumptions
 - Review Results – with team & exit
- 





Source Lines of Code

- Is a size metric
 - Used to estimate efforts (& Productivity?)
 - Effective Lines of Code
 - $eLOC = LOC - (\text{comments line} + \text{Blank Line} + \text{Parenthesis Lines})$
 - Need to generate baseline data for different programming environment
 - Tools easily available to automate SLOC counts
- 
- 

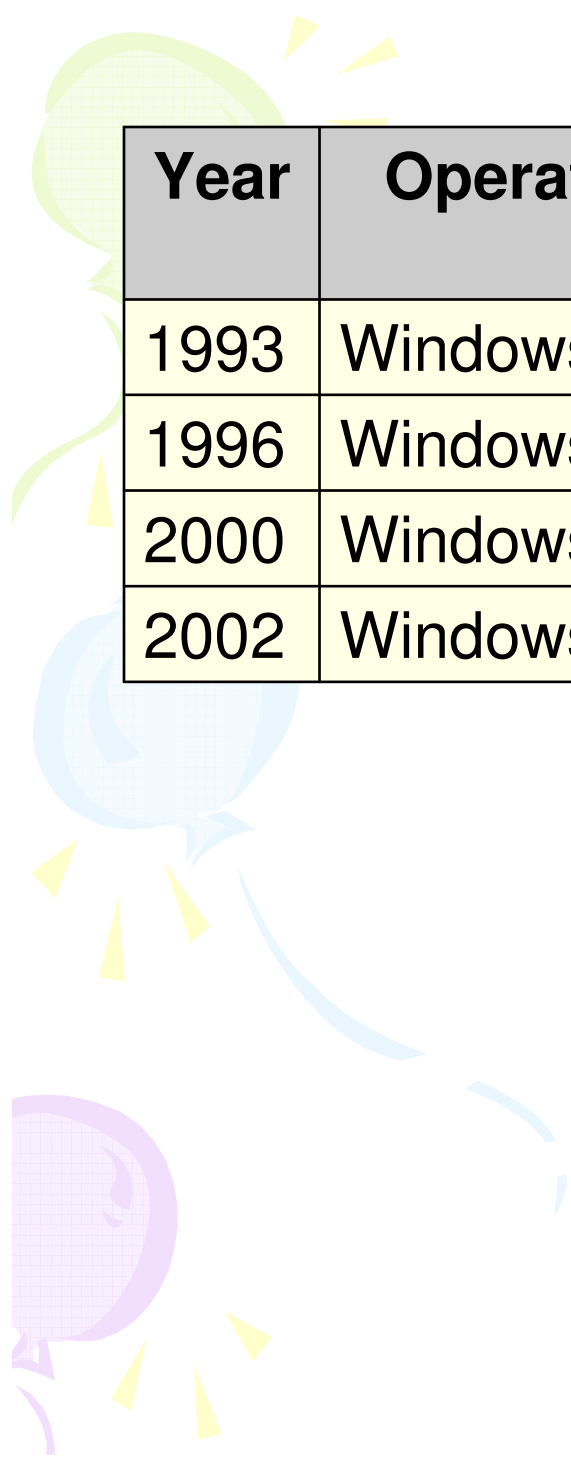


SLOC – is it a good measure

- No Standard definition of LOC
- Advanced IDE – code generation
- Use of Multiple of Language in one software program
- Developer's Experience

```
while ((i=2) < (z= z >>1)) y = z;  
// some other simple lines of code
```

- Hard to predict before code is written
- 





| Year | Operating System | SLOC (Million) |
|-------------|-------------------------|---------------------------|
| 1993 | Windows NT 3.1 | 6 |
| 1996 | Windows NT 4.0 | 16 |
| 2000 | Windows 2000 | 29 |
| 2002 | Windows XP | 40 |

| Operating System | SLOC (Million) |
|-------------------------|-----------------------|
| Red Hat Linux 6.2 | 17 |
| Red Hat Linux 7.1 | 30 |
| Debian 2.2 | 56 |
| Debian 3.0 | 104 |
| Debian 3.1 | 213 |
| Sun Solaris | 7.5 |

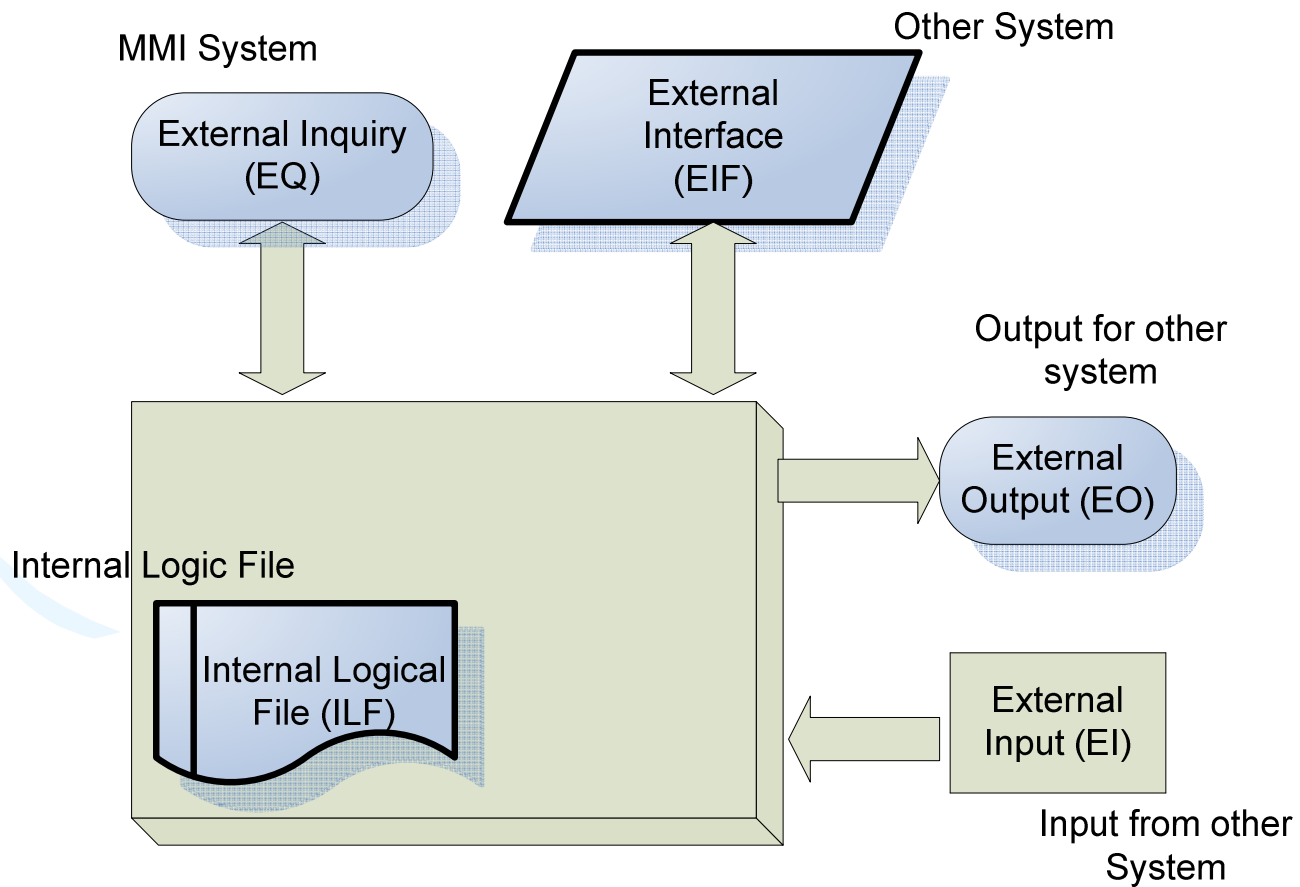
Source - Wikipedia.



Function Point Analysis


- Developed at IBM in 1977 to Measure the Functional Size of an Information System
 - In 1986 IFPUG was formed
 - Based on functionality (value) being developed for the end user
 - Well developed framework for counting the functionalities
- 
- 

FP Framework






FP Framework

- Inputs : Instance through which the user can alter a program's data.
 - Outputs : Instance generated by the program for use by the user or another program.
 - Inquiries: The combination of an input and its resulting output. Outputs are inquiries requiring processing and possibly sophisticated formatting.
 - Logical Internal Files: Logical blocks of data or information unreachable by the user belonging solely to the program.
 - External Interface Files: Logical blocks of data originating from or addressed to other programs (not the user).
- 



FP Steps

- ⊕ Identify the functions of the system that are relevant to the user
 - ⊕ Determine the functional complexity of each function
 - ⊕ Calculate the unadjusted function point count of the system
 - ⊕ Rate the general requirements for the system using the 14 general system characteristics
 - ⊕ Calculate the adjusted function point count of the system
- 

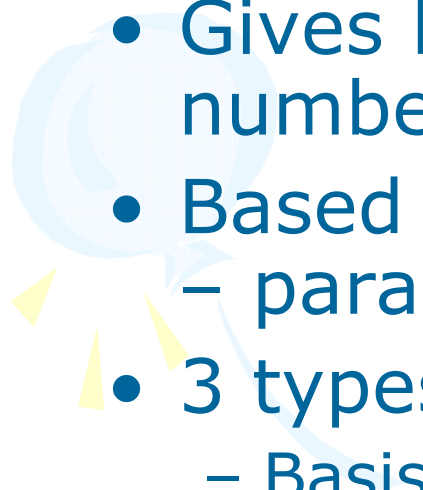
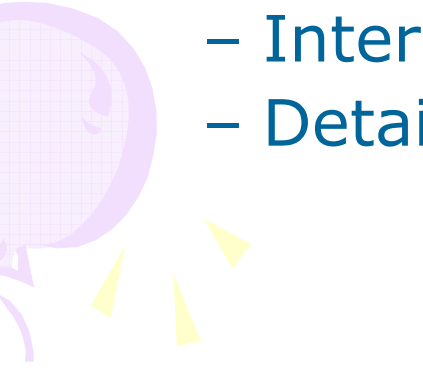


FP Pitfalls

- It determines size not effort. Need historical data to derive effort. Effort differ from developer to developer.
- Manual and time consuming process. IFPUG user manual is huge.



Constructive Cost Model

- Developed by Berry Boehm in 1981
 - Also known as COCOMO81
 - Gives Effort Equation to estimate the number of Person-Months
 - Based on Delivered Source Lines of Codes
 - parametric non linear model
 - 3 types
 - Basis
 - Intermediate
 - Detailed
- 
- 



Constructive Cost Model

- Basic COCOMO

- is a static, single-valued model that computes software development effort (and cost) as a function of program size expressed in estimated lines of code

- Effort = $a * (\text{KLOC})^b P/M$

- where a and b depends on the type of the project



Constructive Cost Model

- Type of Projects

- Organic : relatively small, simple in which small teams with good application experience work to a set of less than rigid requirements.
- Semi Detached : intermediate in size and complexity in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements.
- Embedded - projects that must be developed within a set of tight hardware, software, and operational constraints.



Constructive Cost Model

| | a | b |
|---------------|-----|------|
| Organic | 2.4 | 1.05 |
| Semi Detached | 3.0 | 1.12 |
| Embedded | 3.6 | 1.2 |



Constructive Cost Model

- Basic COCOMO is simple but accuracy is limited as personal experience, tools and techniques and project attributes are not considered.
- Intermediate COCOMO
 - Extension of Basic COCOMO with Project attributes
 - Effort = $EFA * a * (KLOC)^b$ in P/M where EFA is Effort Adjustment Factor. "a" and "b" are constants



Constructive Cost Model

- Intermediate COCOMO
 - EFA is derived from 15 different project attributes grouped in 4 categories
 - Product Attributes – reliability, complexity ...
 - Hardware Attributes – Run time performance, Memory constrains ...
 - Personal Attributes – Analysts capability, Programming experiences ...
 - Project Attributes – Use of tools, Use of Methods, Requirement Schedule.



Constructive Cost Model

- Detailed COCOMO

- Extension of Intermediate COCOMO

- Introduces 2 more parameters

- Phase Sensitive Effort Multipliers –Some phase requires additional attention.

- Three level product hierarchy : The software product is estimated in the three level hierarchical decomposition. The 15 cost drivers are related to module or subsystem level.



COCOMO II

- Extension of COCOMO81
- Updated for Current Development Model – Incremental, Iterative
- Takes care of Component Reuse
- Requirement volatility is considered
- Development Phases are considered instead of Organic, Semi-Detached & Embedded



COCOMO II

- 17 Cost Drivers

| | Post-Architecture Cost Drivers |
|-------------------|--|
| Product Factors | Required Reliability (RELY) |
| | Database Size (DATA) |
| | Product Complexity (CPLX) |
| | Developed for Reusability (RUSE) |
| | Documentation Match to Life-Cycle Needs (DOCU) |
| Platform Factors | Execution Time Constraints (TIME) |
| | Main Storage Constraint (STOR) |
| | Platform Volatility (PVOL) |
| Personnel Factors | Analyst Capability (ACAP) |
| | Programmer Capability (PCAP) |
| | Personnel Continuity (PCON) |
| | Applications Experience (APEX) |
| | Platform Experience (PLEX) |
| Project Factors | Language and tool Experience (LTEX) |
| | Use of Software Tools (TOOL) |
| | Multisite Development (SITE) |
| | Required Development Schedule (SCED) |


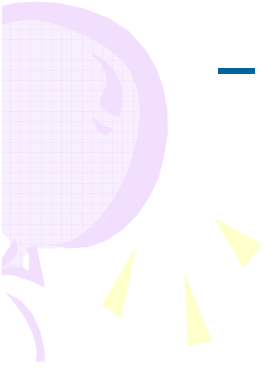


Wrap Up

- Understand the work
 - make sure that you are estimating the right thing
- Understand the process you used
 - Describe how you came up with your estimate, step1, step2 ..



Wrap Up

- The estimating technique(s) you used
 - If possible have multiple techniques to backup your number
 - Analogy, Expert Opinion, Modeling
 - The actual estimate of the effort, duration and cost, if applicable
 - Present your estimate now
- 
- 



Wrap Up

- The assumptions you made in developing the estimate
- Mention the confidence factor
 - 90% confident
- The detailed estimating information
 - Back up your number with details such as WBS



For further study

- Game Theory in Sizing

- http://www.thomsettinternational.com/main/articles/hot_games.htm

- Stop Promising Miracle

- <http://www.processimpact.com/articles/delphi.html>

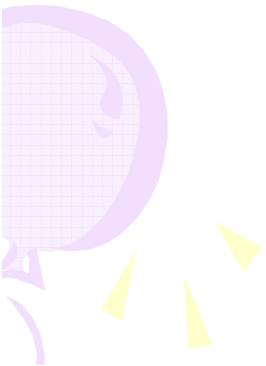
- Function Point Analysis


- <http://www.nesma.nl/english/menu/frsfpa.htm>

- COCOMO

- <http://www.softstarsystems.com/>

- http://en.wikipedia.org/wiki/Intermediate_COCOMO





"One cannot calculate the precise future motion of a particle, but only a range of possibilities for the future motion of the particle."

- Heisenberg, in uncertainty principle paper 1927.