



## What Does Functional Size Measurement Mean?

When Grant Rule was working with early pioneers of agile methods back in the late 1980s, estimating was a big issue. No-one, it seemed, knew how to determine accurately in advance how big a project was, how long it would take, or how much it would cost. This was particularly a problem for a software house delivering fixed price contracts. Focusing on rapid, iterative delivery to ensure the user gets the right software product is good; but at what price?

While advocates of Lean-Agile rightly emphasize the importance of delivering the right thing, it is essential that professional software managers also keep reliable accounts of software process performance for estimating costs, auditing performance, and improving productivity. Effectiveness is *delivering optimum value* for the *minimum cost*.

The solution to Grant's estimating issue, and many associated software project control issues, was the IFPUG function point measurement method developed by Alan Albrecht in 1979. Yet for various reasons, functional size measurement has failed to become widely adopted as a software project management tool. Perhaps it seems arcane - difficult - part of that top-heavy bureaucracy development teams want to be rid of. But as a result, estimating remains a widespread problem for Agile and non-Agile developers alike. Process performance remains hugely variable and unpredictable, with measurement often still the most neglected area of process improvement. By stripping away much of the dysfunctional and redundant process control that can accumulate around the software process, the current interest in Agile is bringing many of these issues to the surface. But many IT professionals seem stuck in a rut, ploughing new land with a newly Agile team but using the same blunt plough and running into the same obstacles.

CIOs, bid teams, development teams, vendor managers and legal advisors seeking better ways of contracting - particularly for Lean-Agile software development - are still faced with the dilemmas Albrecht was seeking to tackle. How do you know your development teams - in-house or outsourced - are delivering best value for money? How does a supplier manage the risk of a fixed price contract? How does he demonstrate competitive productivity levels and on-going improvement? How will Agile benefit *my* organisation? How does the business manage the risk if we can't detail the requirements up-front? How do you measure the benefits of outsourcing IT? Which technology should we choose? Will a change of technology affect the value delivered? What are the relative merits of buying it, building it or outsourcing it?

Alan Albrecht's answer to addressing these issues was to measure software size in terms of the output delivered to the user - its *functionality* - rather than simply in terms of the time and effort put into the development. Deriving a software size from the user's requirements which can then be mapped to the time, resources and budget needed to realise those requirements gives you the necessary objectivity to tackle the scope, project and performance control issues. It also takes you at least part way to measuring the business value and benefit delivered by software. At least you know how much it will cost and how long it will take to deliver the specified functionality; the software should do what it "says on the tin" and it should be delivered on schedule, within budget. If the customer has bought beans instead of tomatoes, or needed a three course meal and not a tin of beans at all, there is something amiss elsewhere in the process.

Functional size measurement means the ability to make objective performance comparisons across teams, across market sectors, and between different suppliers. It means reliable productivity and pricing benchmarks. It means accurate baselines and measures of improvement. It means early and accurate estimations of software project cost and duration. That was why Grant Rule first adopted the technique back in the 1980s - and remains a knowledgeable advocate of it today.



## Estimating\*

*\*Papers referred to below can be found in the Library section at [www.sms exemplar.com/library](http://www.sms exemplar.com/library)*

The predictability of software projects remains a major issue for many business managers. Agile can exacerbate this if delivered software products or software product components have to be routinely re-factored. Customers considering outsourced Agile development will rightly require some assurance of value for money, and suppliers need a reliable oversight of costs to ensure prices remain competitive. The 'Story Point' measures used by many Agile teams to estimate and manage projects are little more than a new take on our old friend, the measure of input. Estimates using Story Points rely on team knowledge and capability, based on individual experience. The process of arriving at them is even known as Planning Poker, an indication of the level of personal skill and informed guesswork required. Story Points cannot be compared between one team and another (let alone one market sector or one supplier and another). Their use either as an effective and reliable estimating tool or as objective and comparable measures of productivity is therefore very limited.

In 2010, Grant Rule published a short paper on using the most modern of the functional size methods - COSMIC - to derive accurate estimates from User Stories (used by Agile teams to capture requirements.) As measurement specialists, SMS recommend the use of COSMIC as the quickest, most reliable, and most easily learned method of introducing robust software estimating practice for developers using Agile or more traditional development practices.

## Size Matters - Accurate Early Estimating

Project failure rates have been shown to relate strongly to the size of project undertaken, with the largest projects accounting for the highest failure rate. Rule's Relative Size Scale was a table developed by Grant Rule to provide a quick and easy early range estimate of project size and cost in terms of clothing sizes - Small, Medium, Large, Extra Large. The scale uses benchmark data to provide a valuable and reliable early-stage feasibility check.

## COCOMO II Profiling - Accounting for Non-Functional Requirements

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry Boehm. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics. COCOMO enables more refined estimates to be derived from simple unadjusted function points by taking into account differences in non-functional project attributes (Cost Drivers). Detailed COCOMO can also account for the influence of individual project phases.

COCOMO II was brought out in 2000, to adapt the original model for use estimating software projects in modern development conditions such as incremental development and rational unified process. COCOMO II is continually evolving to adapt to changing software methods.

## What are Function Points?

Function Points measure the end users requirements in terms of data movements. This means that an early functional size estimate can be derived before any code has been written. They are then also used to measure the actual cost of developing the requirements, which can be compared directly to the estimate and used to calibrate a scale for a particular development environment. If development productivity figures are known, the cost of developing new software can therefore be estimated early



enough to make direct comparisons to the cost of buying a software package, or simply using a non-technical solution. Function Point measures are independent of technology so can compare different suppliers, technologies and different teams to give an objective basis for a cost benefit decision.

Because they encompass both cost (development effort consumed) and benefit (requirements fulfilled), Function Points unlock the door to measuring productivity; velocity; quality; and value in software development.

- The capacity to measure the amount of **output produced for input provided** gives an objective **productivity** figure.
- Measured against time - Eg **FP delivered per elapsed month** - function points provide objective and comparable measures of **velocity**.
- The number of defects detected over total size of delivery - ie **no of defects delivered over no of fp delivered** - gives a measure of **quality**.
- By comparing **costs, effort and time expended per function point delivered**, we can also demonstrate the huge waste generated by ineffective software processes compared to the standards of measurable efficiency shown by **effective**, Right-shifted organisations. Although a long way from being a comprehensive measure of value delivered to the customer, this offers a measure of the component of value for which the software developer is solely responsible. Ensuring the functionality delivered is aligned to business need is the joint responsibility of the business user and the development team and outside the scope of functional size measurement.

### A Comparison of Functional Size Methods

The most widely used function point methods are IFPUG, NESMA, Mk II and COSMIC. Of these, IFPUG is based on the original method designed by Alan Albrecht. COSMIC is the newest method, developed by a international team of metrics experts to support modern software.

Where organisations have already made some investment in IFPUG capability, it may makes sense to use this more widely known method. It is actively managed and maintained by the International Function Point User Group, with the latest version 4.3 released in January 2010. The Certified Function Point Specialist qualification is a recognised standard for IFPUG counting competency. There is considerable IFPUG benchmarking data available, although it is questionable how valuable some of the older data is in terms of benchmarking modern software performance.

While the older functional size measures - typically, IFPUG - can be used for Agile projects, there are significant problems. IFPUG counting is not the easiest process to learn or apply, and even with trained practitioners, time can be wasted debating fuzzy areas. Interpretation of the IFPUG counting rules for complex modern software environments can be tricky, sometimes resulting in differences of opinion between supplier and customer.

COSMIC offers all the advantages of functional size measurement, without some of the shortcomings of earlier methodologies which were not designed with 21<sup>st</sup> century software in mind. It will prove easier to introduce, easier for both business users and software developers to understand and apply, and a more effective communication mechanism for negotiating the delivery of value. It is an established standard with recognised training qualifications. There is a growing repository of benchmark data available.

See the table below for a more detailed comparison.



## Types of measurement scale and permissible operations using them

The type of scale depends on the nature of the relationship between values on the scale. Four types of scale are commonly defined:

- **Nominal** – arbitrary labels, classification data, no ordering – the measurement values are categorical but it makes no sense to state that one category is ‘greater than’ another. For example: Yes/No; Black/White/Yellow/Red; male/female, animal/vegetable/mineral; the classification of defects by their type.
- **Ordinal** – ordered but differences between values are not important – the measurement values are rankings. For example: restaurant ‘star’ ratings; political parties on left to right of the spectrum are given labels Red, Orange, Blue; Likert scales that rank ‘user satisfaction’ on a scale of 1..5; the assignment of a severity level to defects.
- **Interval** – ordered, constant scale, but no natural zero – the measurement values have equal distances corresponding to equal quantities of the attribute. For example: dates, temperature on Celsius or Fahrenheit scales – differences make sense, but ratios do not (e.g.,  $30^{\circ}-20^{\circ} = 20^{\circ}-10^{\circ}$ , but  $20^{\circ}$  is not twice as hot as  $10^{\circ}$ ! Other examples: cyclomatic complexity has the minimum value of one, but each additional path increments the count by one.
- **Ratio** – ordered, constant scale, natural zero – the measurement values have equal distances corresponding to equal quantities of the attribute where the value of zero corresponds to none of the attribute. For example: height; weight; age; length; temperature on Kelvin scale (e.g. absolute zero =  $0^{\circ}\text{K}$ , and  $200^{\circ}\text{K}$  is twice as hot as  $100^{\circ}\text{K}$ ); the size of a software source listing in terms of Non-Commentary Source Statements (or Source Lines Of Code).

The method of measurement usually affects the type of scale that can be used reliably with a given attribute. For example, subjective methods of measurement usually only support ordinal or nominal scales.

Only certain operations can be performed on certain scales of measurement. The following list summarizes which operations are legitimate for each scale. Note that you can always apply operations from a 'lesser scale' to any particular data, e.g. you may apply nominal, ordinal, or interval operations to an interval scaled datum.

- **Nominal Scale.** You are only allowed to examine if a nominal scale datum is equal to some particular value or to count the number of occurrences of each value. For example, gender is a nominal scale variable. You can examine if the gender of a person is F (female) or to count the number of Ms (males) in a sample. Valid statistics: mode, chi square.
- **Ordinal Scale.** You are also allowed to examine if an ordinal scale datum is less than or greater than another value. Hence, you can 'rank' ordinal data, but you cannot 'quantify' differences between two ordinal values. For example, political party is an ordinal datum with the Liberal Democratic Party to the left of the Conservative Party, but you can't quantify the difference. Another example are preference scores, e.g. ratings of eating establishments where 10=good, 1=poor, but the difference between an establishment with a 10 ranking and an 8 ranking can't be quantified. Valid statistics: mode, chi square, median, percentile.
- **Interval Scale.** You are also allowed to quantify the difference between two interval scale values but there is no natural zero. For example, temperature scales are interval data with 25C warmer than 20C and a 5C difference has some physical meaning. Note that 0C is arbitrary, so that it does not make sense to say that 20C is twice as hot as 10C. Valid statistics: mode, chi square, median, percentile, mean, standard deviation, correlation, regression, analysis of variance.
- **Ratio Scale.** You are also allowed to take ratios among ratio scaled variables. Physical measurements of height, weight, and length are typically ratio variables. It is now meaningful to say that 10 metres is twice as long as 5 metres. This ratio holds true regardless of which scale the object is being measured in (e.g. metres or yards). This is because there is a natural zero. Valid statistics: mode, chi square, median, percentile, mean, standard deviation, correlation, regression, analysis of variance, geometric mean, harmonic mean, coefficient of variation, logarithms.



A comparison of the most common Function Size Measurement (FSM) Methods

General Information	IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
Origin	Created by Allan Albrecht at IBM in 1978  Latest release (January 2010) of the original method	Believed to have been created by NESMA (aka NEFPUG) in mid-1980s  Derived from IFPUG	Created by Charles Symons at Nolan Norton in 1984 (put into public domain 1991)  Updated method for use with DBMS, structured methods, CASE tools, etc	Created by international consortium of industry subject matter experts and academics from 19 countries in 1997  Updated method for use with OOA/D, layered architectures, Web2.0, lean/agile, etc
Counting Practices Manual	Available to IFPUG members	Available for sale	Available - public domain	Available - public domain
Counting Practices Manual - languages available	English & some other language versions available to members	Dutch-language version English-language version	English-language version	9 language versions: Arabic, Chinese, Dutch, English, French, German, Italian, Japanese, Spanish
Used by	Public & private sector organisations, large & small, both customers & vendors, around the world  Mostly MIS users Stable user base – international	Public & private sector organisations, large & small, both customers & vendors, primarily in The Netherlands  Mostly MIS users Declining user base – mostly The Netherlands	Originally HM Government’s preferred method for sizing & estimating software. Now used by a few public sector customers & their vendors  Declining user base – mostly United Kingdom	Public & private sector organisations, large & small, both customers & vendors, around the world  MIS and Engineering users Growing user base – international
Terminology used	Founded in the 1970s	Founded in the 1970s	Uses structured methods terminology	Compatible with OOA/D, & software eng. principles
Availability	Available only to members of IFPUG (but easy to join organisation)	Public domain – download from NESMA	Public domain – download from UKSMA	Public domain – download from COSMIC
Design Authority (independent of vendors)	International Function Point Users Group (IFPUG)  <a href="http://www.ifpug.org">www.ifpug.org</a>	Netherlands Software Metrics Association (NESMA)  <a href="http://www.nesma.nl">www.nesma.nl</a>	United Kingdom Software Metrics Association (UKSMA)  <a href="http://www.uksma.co.uk">www.uksma.co.uk</a>	Common Software Measurement International Consortium (COSMIC)  <a href="http://www.cosmicon.com">www.cosmicon.com</a>



**Common Features**

1. Compliance

All four methods comply with the international standard for Functional Size Measurement Methods – ISO14143:

IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
ISO/IEC 20926:2003 ISO Standard applies only to unadjusted FP	ISO/IEC 24570:2005 ISO Standard applies only to unadjusted FP	ISO/IEC 20968:2002 Recommended method for HM Government (UK)	ISO/IEC 19761:2003/2010 BCS Technology Award Winner in 2006 Recognised as a National Standard in Spain & Japan

2. Certification

All four methods operate certification schemes for training measurement staff:

IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
Yes Certified Function Point Specialist (CFPS)	Uses IFPUG CFPS	Yes Certified Function Point Analyst (CFPA)	Yes COSMIC Practitioner Certification

3. Benchmarking Data

All four methods are supported by the International Software Benchmarking Standards Group (ISBSG). There are differences, however, in the size of the comparative data pool:

IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
Large data set compiled over many years – the utility of antique data is questionable	Large Comparisons use IFPUG data	Small Some native data; can be compared to IFPUG data if care is taken	Moderate and growing Data since 1997; ISBSG benchmark released 2009; can be compared to older data if care is taken



4. All four methods share the following characteristics:

- Oriented toward user-required functionality
- Helps verify consistency & completeness of user-required functionality
- Analyses can be used as basis for construction of tests independent of code & test activities
- Measures functional size of dynamic (behavioural) aspects of system (expressed as e.g. use cases, conversational dialogues, user stories, epics & themes, etc)
- Measures development of new requirements
- Measures adaptive maintenance (enhancements)
- Designed for MIS systems - flat & indexed files, batch systems, OLTP systems
- Can be used to measure Functional User Requirements before design, code & test
- Can be used to measure Functional User Requirements after design, code & test
- Can be used to (re)estimate during product life-cycle
- Size can be used as input into top-down software cost models such as COCOMO.II.2000, SLIM, SEER, Price-S, etc
- Can be used to construct product burndown charts, calculate takt time, #sprints, etc
- Independent of product non-functional requirements
- Independent of project constraints
- Independent of developer experience
- Independent of process, project management & development methods
- Early estimates of functional size can be made based on incomplete knowledge of Functional User Requirements – enabling consistent use of one size scale for estimating & measurement throughout project:

IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
Can produce early estimates using various methods: e.g. Fast Eddy, File-Based Approach, Transaction-Based approach	Can produce early estimates using various methods: e.g. Fast Eddy, File-Based Approach, Transaction-Based approach	Can produce early estimates using various methods: e.g. Data Model Approach (CRUDL), Transaction-Based approach	Can produce early estimates using various methods: Event-Based Approach, Object-Based Approach, Story-Based Approach

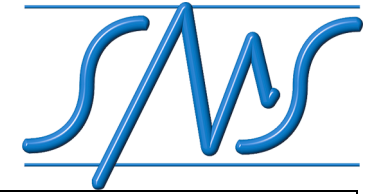


NONE of the four methods deliver the following characteristics:

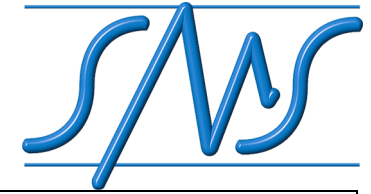
- Measures corrective maintenance (fixes)
- Measures perfective maintenance (refactoring for improved performance)
- Measures algorithmic complexity
- Measures reuse of code

5. Differences between the four main methods:

Characteristic	IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
Measures functional size of static (data storage) aspects of a system (expressed as files, tables, entity types, classes, etc)	Yes	Yes	Regarded as ‘double accounting’ only information processing measured	Regarded as ‘double accounting’ only information processing measured
Compatible with modern methods of requirements analysis	Partially (1975/85s concepts) requires data model	Partially (1980/85s concepts) requires data model	Yes (1980/95s concepts) requires data model	Yes (1995/2010s concepts) incl. incremental
Designed for MIS systems - Relational DBMS	No But mapping rules have been developed	No But mapping rules have been developed	Yes	Yes
Designed to be applicable to real-time and/or embedded systems	No MIS concepts only	No MIS concepts only	No terminology can be re-interpreted for real-time	Yes one common model applicable across MIS, real-time & embedded systems
Can be used to measure complex, layered architectures	No Rules assume monolithic system – infrastructure & middleware is ‘invisible’	No Rules assume monolithic system – infrastructure & middleware is ‘invisible’	Yes Limited – can recognise 3-tier architecture	Yes Designed to recognise ‘layered architectures’ – measures all functional requirements allocated to software systems



Characteristic	IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
<p>Scale type:</p> <p>Nominal – distinguishes members of sets, unordered                      Ordinal – relationship between sets, unequal intervals                      Interval – comparisons, equal intervals, arbitrary zero                      Ratio – comparisons, equal intervals, a natural zero ref: ISO/IEC CD 15939.</p>	<p>‘Nominal/Ordinal’ Scale</p> <p>Unequal intervals between Low &amp; Average, and between Average &amp; High</p>	<p>‘Nominal/Ordinal’ Scale</p> <p>Unequal intervals between Low &amp; Average, and between Average &amp; High</p>	<p>‘Ordinal/Interval’ Scale</p> <p>Weights derived so that 1 MkII fp = 1 IFPUG fp approximately comparing functional processes</p>	<p>Ratio Scale</p> <p>Empirical data suggests 1 cfp = 1 IFPUG fp approximately comparing functional processes</p>
<p>Permissible arithmetic &amp; statistical operations</p>	<p>Categories assigned relative weights:</p> <p>Data can be 'ranked', but 'quantifying' differences between values is difficult due to 'cut off' (Low is c. half of High) – ratios are problematic</p>	<p>Categories assigned relative weights:</p> <p>Data can be 'ranked', but 'quantifying' differences between values is difficult due to 'cut off' (Low is c. half of High) – ratios are problematic</p>	<p>Ordered, synthetic scale with a natural zero:</p> <p>Data can be ranked; differences &amp; ratios between values can be quantified within limits but are problematic due to the use of weights</p>	<p>Ordered, constant scale with a natural zero:</p> <p>Data can be ranked; differences between values can be quantified; ratios make sense (i.e. 20 is twice the size of 10, and 2000cfp is twice 1000cfp).</p>
<p>Accounts for information processing by:</p>	<p>Sizing static data and dynamic behaviour</p>	<p>Sizing static data and dynamic behaviour</p>	<p>Sizing dynamic behaviour, the use of data</p>	<p>Sizing dynamic behaviour, the use of data</p>
<p>Models the functional user requirements as:</p>	<p>File Types and Elementary Process (= Input-Process-Output)</p>	<p>File Types and Elementary Process (= Input-Process-Output)</p>	<p>Logical Transactions (= Input-Process-Output)</p>	<p>Functional Processes (= Input-Process-Output)</p>
<p>Equivalent of stimulus/response message pair (i.e. a 'thread of control with some input, related processing, and some output')</p>	<p>Elementary Process either: External Input (EI), External Output (EO) or External Query (EQ) depending on 'primary intent'</p>	<p>Elementary Process either: External Input (EI), External Output (EO) or External Query (EQ) depending on 'primary intent'</p>	<p>Logical Transaction (LT)</p> <p>All stimulus/response message pairs regarded at LT irrespective of 'primary purpose'</p>	<p>Functional Process (FP)</p> <p>All stimulus/response message pairs regarded at FP irrespective of 'primary purpose'</p>
<p>Rules for measuring size</p>	<p>Different rules apply depending on elementary process type</p>	<p>Different rules apply depending on elementary process type</p>	<p>Same rules apply to all logical transactions</p>	<p>Same rules apply to all functional processes</p>
<p>Base Functional Component(s)</p>	<p>Internal Logical File                      External Interface File                      External Input                      External Output                      External Query</p>	<p>Internal Logical File                      External Interface File                      External Input                      External Output                      External Query</p>	<p>Input Data Element                      Entity Reference                      Output Data Element</p>	<p>Data Movement</p> <p>(either: Entry, eXit, Read, or Write depending on direction of movement)</p>



Characteristic	IFPUG FPA r4.3	NESMA FPA v2.0	Mark II FPA r1.3.1	COSMIC FSM r3.0.1
Contributors to functional size	Per File Type: #static Data Element Types & #Record Element Types  Per Transaction Type: #dynamic Data Element Types & #File Type References	Per File Type: #static Data Element Types & #Record Element Types  Per Transaction Type: #dynamic Data Element Types & #File Type References	Per Logical Transaction: #Input Data Elements #Entity References #Output Data Elements	Per Functional Process: #Data Movements  i.e. the movement (Entry, eXit, Read or Write) of one Data Group
Unit of measure	Different weights assigned to 5 function types depending on their relative 'complexity'  Unit = 1 fp (IFPUG)	Different weights assigned to 5 function types depending on their relative 'complexity'  Unit = 1 fp (NESMA)	Weights assigned to the 'minimum size logical transaction' add to 2.5 to establish comparability between MkII and IFPUG  Unit = 1 fp (MkII)	1 Data Movement = 1 COSMIC Function Point  Unit = 1 cfp
Sensitivity to small changes to requirements	Low  (only detects changes at boundaries between Low, Average, High categories)	Low  (only detects changes at boundaries between Low, Average, High categories)	High  (detects changes of single data element types and single entity references)	Moderate  (detects changes to single data- groups)
Integrity of measures (how well do the measures reflect the thing measured?)	Artificial limits (weights, thresholds, uneven intervals) limit size of function types measured. Integrity is limited.	Artificial limits (weights, thresholds, uneven intervals) limit size of function types measured. Integrity is limited.	No artificial limits imposed on size of functional process.  Integrity is good.	No artificial limits imposed on size of functional process.  Integrity is excellent.
Sensitivity to variation in functional size of dynamic model of system i.e. functional processes	Stepped: minimum step 3fp maximum step 7fp	Stepped: minimum step 3fp maximum step 7fp	Stepped: minimum step either 0.26, 0.58 or 1.66 maximum step infinity	Accommodates size variation from zero to infinity in steps of 1 cfp
Sensitivity to variation in functional size of static model of system i.e. data stores	Stepped: minimum step 5 fp maximum step 15 fp	Stepped: minimum step 5 fp maximum step 15 fp	Data stores are considered to deliver functionality only when the data is referenced in transactions	Data stores are considered to deliver functionality only when the data is used in functional processes
Smallest feasible functional process	3 fp	3 fp	2.5 fp	2 cfp
Smallest feasible enhancement	3 fp	3 fp	0.26 fp	1 cfp