初中級プロマネのための
現場で活かせ！統計情報

土屋俊樹
（株式会社ハイマックス　第３事業本部　シニアコンサルタント）

はじめに

昨年 10 月にソフトウェア開発データ白書の最新版（2016-2017）(*1)が発表された。より情報が豊富になり、新たな分析情報も加えられ非常に充実した内容となっている。これだけの豊富な内容にもかかわらず IPA のサイトでは無償でダウンロードも可能となっている。関係者の方々の努力に頭の下がる思いである。

さて、非常に豊富な内容ではあるが、この膨大な統計値の情報量に圧倒されて、ほしい情報がどこに記載されているのか分からず戸惑うエンジニアも多いのではないだろうか。

事実、筆者も周りのエンジニアからも、「あの数値はどこに記載されているのか？」という問い合わせを受けることが少なからずある。

そこで、今回はデータ白書の膨大な情報の中から著者が実際に現場でよく使う統計値、もしくは参考になるであろうと思われる統計値を、その簡単な使い方とともに紹介したいと思う。特に初級から中級プロジェクトマネジャーの方には、開発の際の定量情報として注目し、経験と勘に頼るだけの開発でなく、工学的なアプローチでも開発を進めてほしい。

データ白書を編纂されている方々からすると、おいしい箇所だけイイとこ取りせず細部までしっかり読み込んでほしい、と叱られそうであるが、IT 現場のためと思ってお許しいただきたい。日々現場で懸命に闘っている IT エンジニアに少しでもお役にたてれば幸いである。

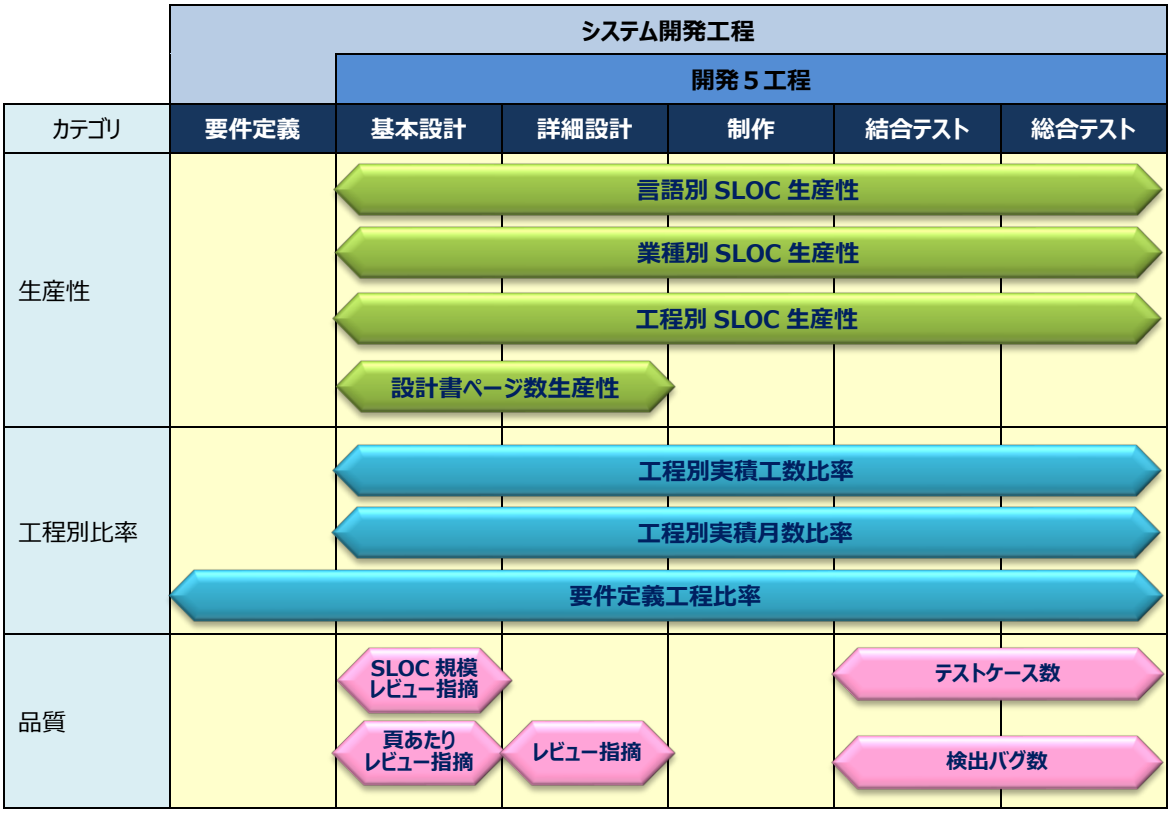
尚、今回紹介する統計値の扱い方については、筆者の理解に基づいて記載しているものである。理解不足による誤用等があれば、その責は筆者にあるのであらかじめご了承願います。

1. 統計値の見方

１－１．統計値の全体像

今回、取り扱う統計情報を、利用目的に応じて大きく①生産性、②工程別比率、③品質にカテゴライズし、システム開発工程にマッピングしたものが下表である。（図 1-1）読者のそれぞれの利用局面において参照されたい。

図 1-1. 取扱い統計情報の全体像



１－２．データ白書の見方

データ白書は統計情報なので基本的に記載例のような形が多い。表の見方を簡単に説明する。

＜記載例＞

図表 8-4-41　主開発言語別 SLOC 生産性の基本統計量（新規開発、主開発言語グループ）　[SLOC／人時]

主開発言語	N	① 最少	② P25	③ 中央	④ P75	⑤ 最大	⑥ 平均	⑦ 標準偏差
COBOL	124	0.533	3.233	5.168	7.557	21.745	5.930	3.850
C 言語	74	0.005	3.078	5.036	8.111	325.239	11.644	37.860
VB	80	0.900	4.245	7.406	11.780	159.965	11.011	18.576
Java	357	0.359	3.135	5.346	8.950	90.868	8.241	10.566

出典：ソフトウェア開発データ白書 2016-2017

＜表の各項目の意味＞

① N　：集計したプロジェクト数。例の場合 Java で 357 プロジェクトの統計値になる。

② 最小：統計値の中の最小値。

③ P25：25 パーセンタイルという。元の数値を昇順に並べて最小値から 25 パーセント目の値。

④ 中央：中央値のこと。数値を昇順に並べて、ちょうど真ん中の値。50 パーセンタイル。
⑤ P75：75 パーセンタイルという。数値を昇順に並べて最小値から 75 パーセント目の値。
⑥ 最大：統計値の中の最大値
⑦ 平均：統計値の平均値
⑧ SLOC／人時：SLOC は Source Lines of Code の略。コメント、空白行を除いたプログラムのステップ数。人時は、一人の 1 時間あたりの工数。人日は 8 人時（1 日 8 時間）、人月は 160 人時（1 ヵ月 20 日）となる。

注目してほしいのは平均値ではなく中央値である。個々の数値のバラツキが大きい場合、平均値より中央値の方が、より実態に近いと思われるからである。Java の場合、中央値は 5.346 なので、1 時間あたり約 5.3 ステップ、1 日あたり約 42.8 ステップ、1 か月あたり約 855.3 ステップの生産性となる。

また、表上に記載されている用語も説明する。

○開発 5 工程　：　基本設計、詳細設計、製作、結合テスト、総合テストのことである。
○新規開発　　：　統計は基本的に新規開発と改良開発に分かれる。本レポートでは主に新規開発を扱う。
○主開発言語グループ　：　開発言語が、COBOL、Java、C、VB のいずれかである。

尚、データ白書にも詳細な解説が記載されているので、より詳しく知りたい方はそちらを参照されたい。

2. 現場での活用法

では、実際の現場ではどのように活用するのか見てみよう。

２－１．見積り時に活用できる統計情報

（1）．生産性

改めて言うまでもなく、開発言語の生産性がわかればシステムの規模（総ステップ数）から全体開発工数（人月）を算出することができる。

全体開発工数（人月）　＝　総ステップ数　÷　生産性（ステップ数／人月）

●言語別 SLOC 生産性

開発言語別の生産性は下表の通りである。数値は人時であることに注意が必要。

図表 8-4-41　主開発言語別 SLOC 生産性の基本統計量（新規開発、主開発言語グループ）　[SLOC／人時]

主開発言語	N	最少	P25	中央	P75	最大	平均	標準偏差
COBOL	124	0.533	3.233	5.168	7.557	21.745	5.930	3.850
C 言語	74	0.005	3.078	5.036	8.111	325.239	11.644	37.860
VB	80	0.900	4.245	7.406	11.780	159.965	11.011	18.576
Java	357	0.359	3.135	5.346	8.950	90.868	8.241	10.566

出典：ソフトウェア開発データ白書 2016-2017

人月に換算した生産性は、下表の通りである。

SLOC 人時→SLOC 人月換算　[SLOC／人月]

主開発言語	N	最少	P25	中央	P75	最大	平均	標準偏差
COBOL	124	85	517	827	1,209	3,479	949	3.850
C	74	1	492	806	1,298	52,038	1,863	37.860
VB	80	144	679	1,185	1,885	25,594	1,762	18.576
Java	357	57	502	855	1,432	14,539	1,319	10.566

新規開発、開発 5 工程（基本設計～総合テスト）での Java の生産性は、855 ステップである。昔から開発現場では、1K ステップ＝1 人月の生産性、と言われているが、データ白書からも、ある程度裏付けのある数値であることがわかる。
尚、この生産性は開発 5 工程での生産性であることに注意したい。単にコーディング、単体テストをするだけならば、1K ステップ＝1 人月では低すぎる。工程別の生産性は後述する。

●業種別 SLOC 生産性

業種別の生産性は下表の通りである。

図表 8-4-37　業種別 SLOC 生産性の基本統計量（新規開発、主開発言語グループ）　[SLOC／人時]

業種	N	最少	P25	中央	P75	最大	平均	標準偏差
製造業	88	0.84	4.68	7.70	12.55	76.12	10.73	10.77
情報通信業	98	0.01	2.88	4.82	7.97	90.87	7.50	10.55
卸売･小売業	55	0.72	4.25	6.14	9.73	82.58	9.02	11.74
金融･保険業	204	0.53	2.54	4.19	6.72	159.97	6.78	13.35
公務（他に分類されないもの）	64	0.41	4.14	7.16	9.02	325.24	12.39	40.01

出典：ソフトウェア開発データ白書 2016-2017

人月に換算した生産性は、下表の通りである。

SLOC 人時→SLOC 人月換算　[SLOC／人月]

業種	N	最少	P25	中央	P75	最大	平均	標準偏差
製造業	88	134	749	1,232	2,008	12,179	1,717	10.77
情報通信業	98	2	461	771	1,275	14,539	1,200	10.55
卸売･小売業	55	115	680	982	1,557	13,213	1,443	11.74
金融･保険業	204	85	406	670	1,075	25,595	1,085	13.35
公務	64	66	662	1,146	1,443	52,038	1,982	40.01

製造業の生産性が高く、金融・保険業の生産性が低いのが興味深い。一般的に金融・保険業は、システム障害発生時の影響が大きいことから、高品質なシステムが求められるため、十分なコストと時間をかける必要がある。よって他業種に比べ生産性が低くなるのだろう。

●工程別 SLOC 生産性

工程別の生産性は下表の通りである。

図表 7-7-2　工程別 SLOC 生産性の基本統計量（新規開発、主開発言語グループ）　[SLOC／人時]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
基本設計	342	0.1	24.4	43.9	85.9	2,855.4	101.6	245.5
詳細設計	342	0.0	20.7	38.0	75.5	4,788.2	94.9	296.9
製作	342	0.0	12.4	20.1	38.3	1,221.1	44.3	110.1
結合テスト	342	0.0	20.5	39.2	73.0	3,314.9	108.0	352.5
総合テスト（ベンダ確認）	342	0.0	29.1	63.8	145.8	66,000.0	500.6	4,115.4

出典：ソフトウェア開発データ白書 2016-2017

人月に換算した生産性は、下表の通りである。

SLOC 人時→SLOC 人月換算　[SLOC／人月]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
基本設計	342	16	3,904	7,024	13,744	456,864	16,256	245.5
詳細設計	342	0	3,312	6,080	12,080	766,112	15,184	296.9
製作	342	0	1,984	3,216	6,128	195,376	7,088	110.1
結合テスト	342	0	3,280	6,272	11,680	530,384	17,280	352.5
総合テスト（ベンダ確認）	342	0	4,656	10,208	23,328	10,560,000	80,096	4,115.4

単に工程毎に生産性を見ると、製作工程では中央値で約 3.2K ステップとなっている。詳細設計まで終わっていれば、コーディング、単体テストで月に約 3K ステップ製造できるということである。設計工程では、製作工程の約 2 倍の生産性がある。基本設計と詳細設計であまり生産性が変わらないことは興味深い。

●設計書ページ数生産性

今回新しく追加された統計値である。小計行を加えた。

図表 7-2-6　SLOC 規模あたりの設計書ページ数の基本統計量（新規開発、主開発言語グループ）　[頁／KSLOC]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
基本設計	113	0.15	3.32	6.09	13.07	54.80	10.08	10.34
詳細設計	113	0.36	5.91	10.72	23.48	611.29	21.90	58.32
			小計	16.81		小計	31.98	

出典：ソフトウェア開発データ白書 2016-2017

1K ステップあたり、基本設計書は約 6 ページ分、詳細設計書は約 10 ページ分の分量になる。この数値により、システム開発規模（総ステップ数）から、設計書のページ分量がある程度見積れるようになるので参考にできる。

例えば、10K ステップ規模の場合、基本設計書は 60 頁、詳細設計書は 100 頁くらいになるということである。

（2）．工程別比率

データ白書には、工程別に工数、および工期の比率の統計値がある。これは中々面白い数値で、ウォーターフォール型開発の場合、開発の各工程でどのような工数配分になっているかわかるので有難い。

●工程別実積工数比率

図表 7-1-13　工程別の実績工数の比率の基本統計量（新規開発）　[比率]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
基本設計	799	0.001	0.103	0.152	0.208	0.589	0.164	0.090
詳細設計	799	0.011	0.121	0.169	0.220	0.533	0.173	0.076
製作	799	0.018	0.264	0.337	0.423	0.847	0.352	0.132
結合テスト	799	0.002	0.120	0.169	0.223	0.588	0.178	0.087
総合テスト	799	0.000	0.071	0.117	0.177	0.564	0.132	0.086
			小計	0.944		小計	0.999	

出典：ソフトウェア開発データ白書 2016-2017

工程別比率では、中央値は 5 工程の合計が 100%にならないので、平均値で見てみよう。比率を図示すると下図のようになる。（図２－１）大きく設計工程、制作工程、テスト工程で括るとおおよそ 3 分の 1 になっている。（図２－２）これは見積りの時の工程別の工数バランスを見るときに参考になるのではないだろうか。

図２－１．工程別工数比率を図示

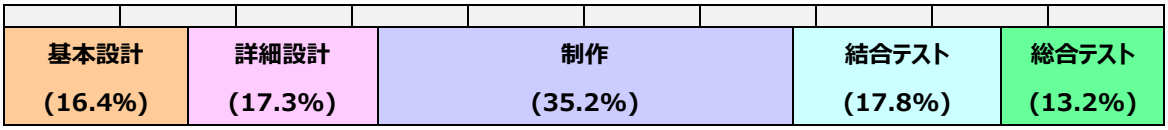
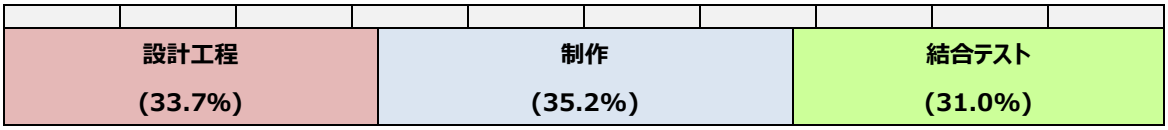


図２－２．設計・制作・テスト工程で区分け



●工程別実積月数比率

次に実績月数つまり工程別にかかった期間の統計値である。

図表 7-1-3　工程別の実績月数の比率の基本統計量（新規開発）　[比率]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
基本設計	309	0.016	0.155	0.201	0.262	0.533	0.216	0.092
詳細設計	309	0.026	0.138	0.184	0.228	0.645	0.189	0.079
製作	309	0.047	0.202	0.248	0.306	0.902	0.262	0.103
結合テスト	309	0.016	0.119	0.165	0.212	0.604	0.172	0.081
総合テスト	309	0.014	0.091	0.149	0.208	0.765	0.162	0.097
			小計	0.947		小計	1.001	

出典：ソフトウェア開発データ白書 2016-2017

図２－３．工程別実績月数比率を図示

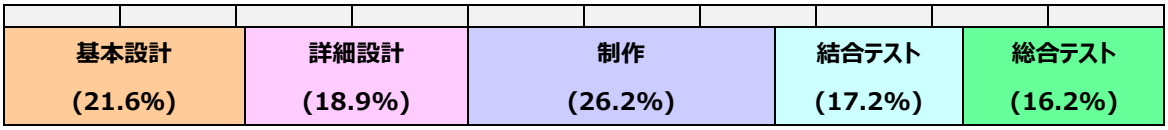
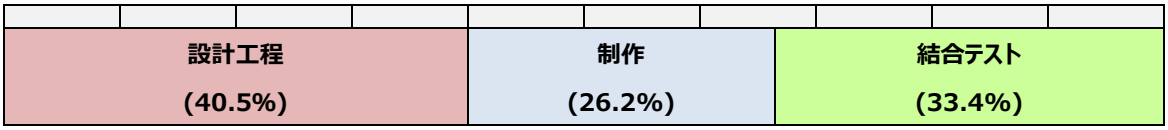


図２－４．設計・制作・テスト工程で区分け



実績工数（工数）と実績月数（工期）を比較すると、設計工程は工数より工期の方の比率が大きく、逆に制作工程はその逆になっている。これは、少ない要員体制で長く時間をかけて設計し、短い期間で一気に製造するということで、一般的な要員の山積みと合致しているのではないだろうか。基本に忠実な開発プロジェクトが多いのだと思う。

●要件定義工程比率

システム開発案件が開始される際、要件定義工程の工数見積りの妥当性を検証することは難しいと思う。要件の難易度によってかなり差が出てくるからだ。データ白書では開発５工程と要件定義工程の比率の統計値がある。参考にすると良いかもしれない。

図表 7-1-15　要件定義工程も含めた工程別の実績工数の比率の基本統計量（新規開発）　[比率]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
要件定義	442	0.001	0.046	0.082	0.130	0.672	0.098	0.076
開発 5 工程	442	0.328	0.870	0.918	0.954	0.999	0.902	0.076
			小計	1.000		小計	1.000	

出典：ソフトウェア開発データ白書 2016-2017

図表 7-1-5　要件定義工程も含めた工程別の実績月数の比率の基本統計量（新規開発）　[比率]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
要件定義	201	0.026	0.112	0.154	0.230	0.743	0.182	0.102
開発 5 工程	201	0.257	0.770	0.846	0.888	0.974	0.818	0.102
			小計	1.000		小計	1.000	

出典：ソフトウェア開発データ白書 2016-2017

工数では、開発５工程に対し約１割程度の要件定義工数が必要であることがわかる。また期間としては中央値で 15%程度の要件定義期間を確保する必要があることがわかる。
筆者が従事する流通系のシステム開発と比較すると、いささか要件定義に工数・期間を掛け過ぎのような気もするが、データ白書で集計されているプロジェクトは、ほぼ成功しているプロジェクトなので(*2)、逆に言えばプロジェクトを成功させるためには、要件定義工程に十分な工数と期間を掛けることが大事なのだということが理解できる。

２－２．品質評価時に活用できる統計情報

品質評価に使える統計値として、主に設計工程ではレビュー指摘数、テスト工程ではテストケース数、検出バグ数が挙げられる。

（１）．基本設計工程

●SLOC 規模あたり基本設計書レビュー指摘数

図表 7-3-2 SLOC 規模あたりの基本設計レビュー指摘件数の基本統計量　[件／KSOLC]

N	最少	P25	中央	P75	最大	平均	標準偏差
436	0.000	0.753	2.381	5.025	300.000	6.396	19.352

出典：ソフトウェア開発データ白書 2016-2017

中央値で基本設計 1K ステップあたり約 2.3 件の指摘である。一般的なウォーターフォール型開発では、設計工程段階ではまだ製造に着手しておらず、1K あたりのレビュー指摘数、と言われても実感しづらいかもしれない。よってより実態に近いページ数あたりのレビュー指摘数の統計値がある。

●ページ数あたり基本設計書レビュー指摘数

図表 7-3-5 ページあたりの基本設計レビュー指摘件数の基本統計量　[件／ページ]

N	最小	P25	中央	P75	最大	平均	標準偏差
270	0.000	0.110	0.281	0.620	10.000	0.580	0.991

出典：ソフトウェア開発データ白書 2016-2017

中央値では、基本設計 1 ページあたり 0.28 件の指摘、つまり 10 ページで 2.8 件のレビュー指摘件数が目安となる。

（２）．制作工程

制作工程でのレビュー指摘数は、SLOC 規模当りのものが欲しいが、回答数が少ないためデータ白書には記載されていない。おそらく大半のプロジェクトは、例えばソースコードレビューであれば、開発対象プログラム全量に対してではなく、サンプリングでレビューしているからであろうか。参考として、工数あたりのレビュー指摘件数を記載する。

●レビュー指摘数

図表 7-3-7 工数あたりの製作工程レビュー指摘件数の基本統計量（2）　[件／160 人時]

N	最小	P25	中央	P75	最大	平均	標準偏差
174	0.0	45.7	121.5	290.3	18623.8	316.9	1420.5

出典：ソフトウェア開発データ白書 2016-2017

（3）．テスト工程

テスト工程では、テストケース数と検出バグ数の統計値がある。数ある統計値の中でもこの数値が現場で一番使われる数値ではないだろうか。この数値自体は、顧客からも求められることが多いと思われる。データ白書ではテストケースと検出バグ数を同表上に記載している。

図表 7-5-16 テスト工程別 SLOC 規模あたりのテストケース数、検出バグ数の基本統計量（ 全開発種別）　[件／KSLOC]

工程	N	最少	P25	中央	P75	最大	平均	標準偏差
結合テスト（テストケース）	901	0.125	16.422	39.022	91.364	63800.000	232.774	2266.010
総合テスト（テストケース）	963	0.017	4.460	12.439	34.721	15200.000	98.561	760.056
結合テスト検出バグ数（現象）	891	0.000	0.528	1.296	2.538	700.000	3.421	24.212
総合テスト検出バグ数（現象）	911	0.000	0.065	0.296	0.854	64.300	1.013	3.357
結合テスト検出バグ数（原因）	413	0.000	0.428	1.175	2.369	700.000	3.849	34.597
総合テスト検出バグ数（原因）	414	0.000	0.063	0.200	0.726	32.066	0.756	2.112

出典：ソフトウェア開発データ白書 2016-2017

●テストケース数

何をもって 1 ケースとカウントするのか、プロジェクトによって基準が違うので、標準偏差が大きい値になっているのだろうか。ともあれテスト計画策定時の目安となる統計値がテストケース数である。中央値でみると、1K ステップあたり、結合テストで約 40 ケース、総合テストで約 12 ケースのテストが必要となっている。

●検出バグ数

中央値で見ると、1K ステップあたり、結合テストで 1.3 件、総合テストで 0.3 件の検出バグ数である。全体の大半のプロジェクト（P25～P75）でも、結合テストでは、0.5 件～2.5 件の範囲内であり、総合テストでは、0.06 件～0.8 件の範囲内となっている。プロジェクト特性を考慮しつつ品質目標としてこの範囲内を目指すべきだろう。

２－３．工程別の成果物量

今回、新たに工程別の成果物量と工数に強い相関関係があるプロジェクトの分析も加えられた。強い相関関係は一般的なウォーターフォール型開発に言えることなので掲載する。

図表 7-8-17 開発規模あたりの成果物量及び成果物量あたりの工数の中央値の一覧（新規開発、主開発言語）

開発工程	要件定義	基本設計	詳細設計	製作	結合テスト	総合テスト(ベンダ確認)
データ数	78	147	141	573	367	345
開発規模当りの成果物量の中央値	KSLOC 当りの要件定義書ページ数（ページ/KSLOC）	KSLOC 当りの基本設計書ページ数（ページ/KSLOC）	KSLOC 当りの詳細設計書ページ数（ページ/KSLOC）		KSLOC 当りの結合テストケース数（ケース/KSLOC）	KSLOC 当りの総合テストケース数（ケース/KSLOC）
	1.27	5.93	12.12		32.58	8.83
成果物量当りの工数の中央値	要件定義書ページ当りの要件定義工数（人時/ページ）	基本設計書ページ当りの基本設計工数（人時/ページ）	詳細設計書ページ当りの詳細設計工数（人時/ページ）	KSLOC 当りの製作工数（人時/KSLOC）	結合テストケース当りの結合テスト工数（人時/ケース）	結合テストケース当りの総合テスト工数（人時/ケース）
人時→	13.76	5.21	2.9	54.03	1.1	2.56
人日→	1.72	0.65	0.36	6.75	0.14	0.32
人月→	0.086	0.033	0.018	0.338	0.007	0.016

出典：ソフトウェア開発データ白書 2016-2017　をもとに人日、人月換算を追記

前述の生産性やケース数の統計値と若干異なるが、ほぼ近い値である。データ白書に記載がある通り(*3)、成果物量がわかっているとき、もしくは工数がわかっているときに、その逆の値を確認するときに使えるかもしれない。

おわりに

データ白書に記述してある通り、統計値の使い方には注意が必要である。(*4)
数値自体をそのまま適用せず、あくまでも参考値として扱い、そこに照らし合わせて自分のプロジェクトの特性を加味して、適用する定量数値を決めるべきであろう。
とは言え、このような統計値にあまり馴染みのない IT エンジニアには、まずはここに紹介した統計値を、自分のプロジェクトと実際に比較してみることをお勧めする。そこで興味をもった、もしくは疑問をもったならば、ぜひデータ白書自体を参照してみてほしい。当レポートで扱っていない情報が満載されている。そしてその情報を、ぜひ自分の現場プロジェクトに活かしてほしい。

さいごに、長年にわたりソフトウェア開発データ白書を発表している IPA、および関係各社、関係者の方々に深く感謝したい。筆者も一エンジニアとして同書を大いに活用させていただいています。

【著者略歴】土屋 俊樹（つちや　としき）
株式会社ハイマックス　第３事業本部第３部　シニアコンサルタント
主に大手流通小売業向けシステム開発にプロジェクトマネジャーとして従事。
近年はエンドユーザーシステム部員向けの人材育成研修の企画・講師も担当する。
情報処理技術者（ST、PM、AE、DB、SM、SC）。

参考文献、および脚注

[1]　「ソフトウェア開発データ白書 2016-2017」　情報処理推進機構
https://www.ipa.go.jp/sec/publish/tn12-002.html
[2]　同書　P75　図表 4-14-3　実績の評価（QCD）
[3]　同書　P228　7.8　工程別の成果物量と工数　【備考】
[4]　同書　P34　3.5 白書利用にあたっての注意事項