Università degli Studi di Padova

KaleidosCode

SWEDesigner

Software per diagrammi UML

Piano di qualifica v1.0.0

Informazioni sul documento

Versione	1.0.0
Data Redazione	09/03/2017
Redazione	Bonato Enrico
	Bonolo Marco
	Pace Giulio
	Sovilla Matteo
Verifica	Pezzuto Francesco
Approvazione	Sanna Giovanni
Uso	Esterno
Distribuzione	*Prof. Vardanega Tullio*
	Prof. Cardin Riccardo
	Zucchetti s.p.a.

kaleidos.codec6@gmail.com

Diario delle Modifiche

Versione	Data	Autore	Descrizione
0.2.0	31/03/2017	Pezzuto Francesco	Verifica del documento
0.1.1	28/03/2017	Sovilla Matteo	Correzione e integrazione come indicato da verifica
0.1.0	25/03/2017	Pezzuto Francesco	Verifica del documento
0.0.4	23/03/2017	Sovilla Matteo	Stesura parte capitolo 3
0.0.3	23/03/2017	Bonato Enrico	Stesura parte capitolo 3
0.0.2	22/03/2017	Pace Giulio	Stesura sezione Obiettivi di qualità
0.0.1	09/03/2017	Bonolo Marco	Creazione scheletro del documento e stesura della sezione Introduzione

Indice

1 Introduzione **1**
1.1 Scopo del documento . . . 1
1.2 Scopo del prodotto . . . 1
1.3 Glossario . . . 1
1.4 Riferimenti utili . . . 1
1.4.1 Riferimenti normativi . . . 1
1.4.2 Riferimenti informativi . . . 1

2 Visione generale della strategia **3**
2.1 Obiettivi di qualità . . . 3
2.1.1 Funzionalità . . . 3
2.1.2 Affidabilità . . . 4
2.1.3 Usabilità . . . 4
2.1.4 Efficienza . . . 4
2.1.5 Manutenibilità . . . 5
2.1.6 Portabilità . . . 5
2.1.7 Altre qualità . . . 5
2.2 Organizzazione . . . 5
2.3 Scadenze temporali . . . 6
2.4 Responsabilità . . . 6

3 La strategia di gestione della qualità nel dettaglio **7**
3.1 Risorse . . . 7
3.1.1 Necessarie . . . 7
3.1.2 Disponibili . . . 7
3.2 Misure e metriche . . . 7
3.2.1 Metriche per i processi . . . 8
3.2.2 Metriche per i documenti . . . 8
3.2.3 Metriche per il codice . . . 9
3.2.4 Tabella riepilogativa . . . 12

4 Gestione amministrativa **13**
4.1 Definizione di un errore . . . 13
4.2 Comunicazione degli errori . . . 13
4.3 Risoluzione degli errori . . . 13

Elenco delle tabelle

2 Riepilogo misure e metriche 12

1 Introduzione

1.1 Scopo del documento

Questo documento definisce gli obiettivi e le metodologie che ogni membro del gruppo *KaleidosCode* adotterà per garantire un determinato livello di qualità del prodotto.
A tal proposito ogni membro del gruppo è tenuto a leggere, perseguire e raggiungere gli obiettivi definiti in esso.

1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di un software di costruzione di diagrammi UML_{G} con la relativa generazione di codice Java_{G} e $\text{Javascript}_{\text{G}}$ utilizzando tecnologie web. Il prodotto deve essere conforme ai vincoli qualitativi richiesti dal committente.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite sono riportate nel documento *Glossario v1.0.0*.
Ogni occorrenza di vocaboli presenti nel *Glossario* è marcata da una "G" maiuscola in pedice.

1.4 Riferimenti utili

1.4.1 Riferimenti normativi

- **$\text{Capitolato}_{\text{G}}$ d'appalto**:
  http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6.pdf (09/03/2017);
- **Norme di progetto**: *Norme di progetto v1.0.0*.

1.4.2 Riferimenti informativi

- **Slide dell'insegnamento di Ingegneria del Software 1° semestre**:
  - Qualità del software:
    http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L10.pdf (09/03/2017);
  - Qualità di Processo:
    http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L11.pdf (09/03/2017).
- **Slide dell'insegnamento di Ingegneria del Software 2° semestre**:
  - Metodi e obiettivi di quantificazione:
    http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L15.pdf (25/03/2017).
- **ISO_{G} 9001**:
  https://it.wikipedia.org/wiki/Norme_della_serie_ISO_9000#ISO_9001 (25/03/2017);

- **ISO 9126**: https://it.wikipedia.org/wiki/ISO/IEC_9126 (25/03/2017);
- **ISO 12207**: https://en.wikipedia.org/wiki/ISO/IEC_12207 (25/03/2017);
- **Indice Gulpease$_G$**:
  https://it.wikipedia.org/wiki/Indice_Gulpease (25/03/2017);
- **Complessità ciclomatica**:
  https://en.wikipedia.org/wiki/Cyclomatic_complexity (25/03/2017);
- **Capability Maturity Model (CMM$_G$)**:
  https://en.wikipedia.org/wiki/Capability_Maturity_Model (25/03/2017);
- **Analisi dei requisiti**: *Analisi dei requisiti v1.0.0*;
- **Piano di progetto**: *Piano di progetto v1.0.0*;
- **Glossario**: *Glossario v1.0.0*.

2 Visione generale della strategia

Per garantire la qualità dei prodotti realizzati durante lo sviluppo del progetto è indispensabile definire e perseguire strategie che assicurino la qualità dei processi adottati nonché il loro continuo miglioramento; inoltre, è necessario definire metriche e pianificare attività che valutino in modo preciso la qualità dei prodotti ottenuti e dei processi adottati. A tal scopo verranno adottate le seguenti strategie:

- Definizione accurata di norme che regolamentano e standardizzano i processi coinvolti nel progetto in termini di:
  - Processi di fornitura;
  - Processi di sviluppo;
  - Processi di supporto;
  - Processi organizzativi.
- Descrizione dettagliata delle strategie di pianificazione adottate per lo sviluppo del progetto in termini di:
  - Modello di sviluppo adottato;
  - Analisi dei rischi che si possono incontrare;
  - Pianificazione delle attività e dei tempi;
  - Stima preventiva delle risorse che saranno impiegate;
  - Assegnazione delle risorse al fine di portare a termine le attività pianificate nei tempi previsti;
  - Consuntivo, durante lo sviluppo del progetto, delle risorse impiegate.
- Ad ogni processo coinvolto nello sviluppo del progetto verrà applicato il principio PDCA_{G} affiancato dal modello CMM. Essi permettono il controllo, la valutazione e il miglioramento continuo dei processi nonché la determinazione del livello di maturità dell'organizzazione nel gestirli.

2.1 Obiettivi di qualità

Prendendo come riferimento lo standard [ISO/IEC 9126] e lo standard [ISO/IEC 12207], il gruppo *KaleidosCode* si impegna a garantire che il prodotto *SWEDesigner* abbia le seguenti qualità:

2.1.1 Funzionalità

Si garantisce che il sistema prodotto abbia tutte le funzionalità indicate nel documento *Analisi dei requisiti v1.0.0*. L'implementazione di ogni requisito deve essere quanto più completa ed economica.

- **Misura**: l'unità di misura utilizzata sarà la quantità di requisiti mappati in componenti del sistema create e funzionanti.

- **Metrica**: la sufficienza è raggiunta quando vengono soddisfatti tutti i requisiti obbligatori.
- **Strumenti**: il sistema deve superare tutti i test che saranno previsti in fase di progettazione.

2.1.2 Affidabilità

Il sistema deve essere quanto più possibile robusto. Nel caso di eventuali errori deve essere di facile ripristino.

- **Misura**: l'unità di misura utilizzata sarà la quantità di esecuzioni che vanno a buon fine.
- **Metrica**: poiché non è possibile valutare ancora tutte le casistiche di utilizzo, le esecuzioni dovranno coprire al meglio la gamma di possibilità. Risulta quindi impossibile stabilire oggettivamente una soglia di sufficienza.
- **Strumenti**: da definire.

2.1.3 Usabilità

Il sistema deve coniugare la facilità di apprendimento e utilizzo con il soddisfacimento di tutte le necessità dell'utente.

- **Misura**: vista la mancanza di una metrica oggettiva adatta, l'unità di misura utilizzata sarà una valutazione soggettiva dell'usabilità.
- **Metrica**: non esistendo una metrica oggettiva adatta, è impossibile stabilire una soglia di sufficienza. I membri del gruppo si impegneranno comunque a garantire un'elevata esperienza d'uso.
- **Strumenti**: ci si affiderà ad un piccolo campione di utilizzatori esterni per avere un riscontro quantomeno realistico dell'usabilità. Per maggiori informazioni si vedano *Norme di progetto v1.0.0*.

2.1.4 Efficienza

Il sistema deve ridurre al minimo l'utilizzo delle risorse impiegate e deve fornire le funzionalità richieste nel minor tempo possibile.

- **Misura**: il tempo di latenza dell'editor per eseguire un comando.
- **Metrica**: la sufficienza viene definita come un tempo di latenza inferiore ai 2 secondi.
- **Strumenti**: da definire.

2.1.5 Manutenibilità

Il sistema deve essere comprensibile ed estensibile.

- **Misura**: le unità di misura utilizzate saranno quelle descritte nella sezione 3.2.3.
- **Metrica**: il prodotto deve ottenere la sufficienza in tutte le metriche descritte nella sezione 3.2.3.
- **Strumenti**: consultare sezione "Norme di codifica" in *Norme di progetto v1.0.0*.

2.1.6 Portabilità

Il sistema deve essere più portabile possibile. Il front-end$_G$ deve essere utilizzabile dal maggior numero di browser$_G$ possibili.

- **Misura**: il front-end deve rispettare gli standard del W3C$_G$.
- **Metrica**: il software deve avere le caratteristiche di portabilità descritte. È necessario che il prodotto raggiunga la sufficienza in tutte le metriche descritte nella sezione 3.2.3.
- **Strumenti**: consultare *Norme di progetto v1.0.0*.

2.1.7 Altre qualità

Saranno importanti per il prodotto anche le seguenti proprietà:

- **Incapsulamento**: aumenta il riuso e la manutenibilità del codice. Saranno utilizzate interfacce ove possibile;
- **Coesione**: le funzionalità che hanno gli stessi obiettivi devono risiedere nello stesso componente in modo da favorire semplicità e manutenibilità, oltre a ridurre l'indice di dipendenza.

2.2 Organizzazione

La gestione della strategia di verifica si basa sull'attuazione delle relative attività descritte nelle *Norme di progetto v1.0.0*. Tali attività vengono eseguite per ogni processo attuato allo scopo di verifica della qualità del processo stesso e dell'eventuale prodotto ottenuto facendo riferimento anche alle metriche definite nella sezione 3.2. Ogni documento prevede un diario delle modifiche che permette di concentrare l'attività di verifica solo nelle parti modificate dopo l'ultima eseguita. Data la diversa natura dei prodotti ottenuti dalle fasi del progetto si applicherà, per ognuno di essi, una diversa procedura di verifica:

- **Analisi e Analisi di dettaglio**: in questa fase si effettuerà una prima stesura dei documenti illustrati nel *Piano di progetto v1.0.0*.
  - Verrà controllata la correttezza ortografica con *LanguageTool 3.6*, opportunamente integrato in *TexStudio*;

  - Verrà controllata la correttezza lessicale con un'attenta ed accurata rilettura affiancata dal controllo di *LanguageTool 3.6*;
  - Verrà controllata la correttezza dei contenuti rispetto alle aspettative del documento;
  - Verrà controllata la corrispondenza di ciascun caso d'uso con un requisito, mediante l'utilizzo dell'applicativo web creato appositamente;
  - Verrà controllato il rispetto delle *Norme di progetto v1.0.0* da parte di ciascun documento;
  - Verranno controllate le rappresentazioni grafiche, figure e tabelle assicurandosi che per ciascuna di esse sia presente un'opportuna didascalia;

- **Progettazione architetturale**: verrà controllato che tutti i requisiti corrispondano ad un componente individuato in questa fase e se ne assicurerà la tracciabilità;
- **Progettazione di dettaglio e Codifica**: durante ciascuna delle iterazioni di questa fase si svolgeranno i test per la verifica del codice. Tali attività avverranno nel modo più automatizzato possibile. I *Verificatori* supervisioneranno questa fase controllando la presenza di eventuali errori;
- **Validazione e verifica**: in questa fase verrà effettuato il collaudo del prodotto, in modo da assicurare il suo corretto funzionamento al momento della consegna.

2.3 Scadenze temporali

Dato l'obiettivo di rispettare le scadenze fissate nel *Piano di progetto v1.0.0*, è indispensabile pianificare l'attività di verifica della documentazione e del codice prodotto in modo che risulti sistematica e organizzata. Grazie all'applicazione di tale strategia l'individuazione e la correzione degli errori avverrà il prima possibile, impedendo la loro rapida diffusione e mitigando la possibilità che gli stessi si ripresentino in futuro, diminuendo così il rischio di ritardi. Tale pianificazione è documentata nel *Piano di progetto v1.0.0* il quale contiene, nella sottosezione 1.4, anche le scadenze temporali che il gruppo *KaleidosCode* si impegna a rispettare.

2.4 Responsabilità

I ruoli responsabili delle attività di verifica sono il *Responsabile di progetto* e il *Verificatore*. I loro compiti e responsabilità, descritti nelle *Norme di progetto v1.0.0*, permettono alle attività di verifica di essere efficienti e sistematiche. Ogni componente del team è responsabile del materiale da lui prodotto.

3 La strategia di gestione della qualità nel dettaglio

3.1 Risorse

3.1.1 Necessarie

Per la realizzazione del prodotto sono necessarie le risorse umane e tecnologiche elencate di seguito.

- **Risorse umane**: sono descritte dettagliatamente nel *Piano di progetto v1.0.0*:
  - *Responsabile di progetto*;
  - *Amministratore*;
  - *Analista*;
  - *Progettista*;
  - *Programmatore*;
  - *Verificatore*.
- **Risorse software**: sono descritte dettagliatamente nelle *Norme di progetto v1.0.0*. Si tratta di software che permettono:
  - la comunicazione e la condivisione del lavoro tra gli elementi del team;
  - la stesura della documentazione in formato LaTeX$_G$;
  - la creazione di diagrammi UML;
  - la codifica nei linguaggi di programmazione scelti;
  - la semplificazione delle attività di verifica;
  - la gestione dei test sul codice.
- **Risorse hardware**: ciascun componente del gruppo deve avere un computer con tutti i software necessari descritti nelle *Norme di progetto v1.0.0*. È necessario avere a disposizione almeno un luogo dove poter effettuare le riunioni interne.

3.1.2 Disponibili

Ogni membro del team ha a disposizione uno o più computer personali dotati degli strumenti necessari.
Le riunioni interne si svolgono presso le aule del dipartimento di Matematica dell'Università degli Studi di Padova.

3.2 Misure e metriche

Il processo di verifica deve essere quantificabile per fornire informazioni utili. Bisogna quindi stabilire le metriche da adottare per le misurazioni durante i processi di verifica. Si definiranno due intervalli di misure (range$_G$):

- **Range di accettazione**: intervallo di valori vincolante per l'accettazione del prodotto;

- **Range ottimale**: intervallo di valori entro cui è consigliabile rientri la misurazione. Il mancato rispetto di questa condizione non pregiudica l'accettazione del prodotto, ma richiede verifiche più approfondite in merito.

3.2.1 Metriche per i processi

Schedule Variance$_G$
È una metrica di progetto standard, indica se si è in linea, in anticipo o in ritardo rispetto alla schedulazione pianificata delle attività di progetto. È pari alla differenza tra il valore delle attività realizzate ed il valore delle attività pianificate alla data corrente.

Parametri utilizzati

- **Range di accettazione**: $\geq -(preventivo * 5\%)$;
- **Range ottimale**: ≥ 0.

Budget Variance$_G$
È una metrica di progetto standard, indica se si è speso di più o di meno rispetto a quanto preventivato alla data corrente. È pari alla differenza tra costo pianificato e costo effettivamente sostenuto alla data corrente.

Parametri utilizzati

- **Range di accettazione**: $\geq -(preventivo * 10\%)$;
- **Range ottimale**: ≥ 0.

3.2.2 Metriche per i documenti

Indice Gulpease
Definito nel 1988 all'Università degli Studi di Roma "La Sapienza" per valutare la leggibilità di un documento redatto in lingua italiana, l'indice Gulpease si basa sul calcolo del numero di caratteri contenuto in una parola rapportato con altri fattori quali il numero di parole e di frasi. La formula per il calcolo dell'indice Gulpease è la seguente:

$$89 + \frac{300\,(numero\ di\ frasi) - 10\,(numero\ di\ lettere)}{numero\ di\ parole}$$

Il risultato indica quindi la complessità del documento con un valore compreso tra 0 e 100, dove 100 indica la più alta leggibilità. Attraverso gli studi condotti, risulta che testi con un indice:

- **inferiore a 80** sono difficili da leggere per chi ha la licenza elementare;
- **inferiore a 60** sono difficili da leggere per chi ha licenza media;
- **inferiore a 40** sono difficili da leggere per chi ha un diploma superiore.

Tale indice, però, non indica la comprensibilità del testo. Il documento potrebbe contenere frasi incomprensibili ed avere comunque un alto indice Gulpease. Per la tipologia dei documenti redatti, la formalità nella scrittura e gli argomenti trattati risulta difficile adeguare la stesura del testo ad un indice Gulpease ottimale. Ogni documento sarà quindi controllato anche da un essere umano che avrà il compito di valutare se parti di testo dovranno essere semplificate o meno. Inoltre, i limiti imposti da tale indice saranno sufficientemente rilassati per accettare anche frasi poco più complesse.

Parametri utilizzati

- **Range di accettazione**: 40 - 100;
- **Range ottimale**: 50 - 100.

3.2.3 Metriche per il codice

Rapporto linee di commento su linee di codice
Indica il rapporto tra linee di commento e linee di codice in un file (linee vuote escluse). Ritenendo importante la rapidità di comprensione del codice, questa metrica è utile per stimare la manutenibilità.

Parametri utilizzati

- **Range di accettazione**: ≥ 0.25;
- **Range ottimale**: ≥ 0.30.

Numero di parametri
Indica il numero di parametri formali di un metodo. Più alto è il numero dei parametri formali, più aumenta la quantità di memoria occupata nella pila dei processi.

Parametri utilizzati

- **Range di accettazione**: 0 - 8;
- **Range ottimale**: 0 - 5.

Numero di campi dati
Indica il numero di campi dati interni ad una classe. Un numero elevato può rendere difficile la manutenibilità del codice della classe oltre ad essere indice di cattiva programmazione.
È possibile ridurre il numero di campi dati attraverso l'incapsulamento di ulteriori classi.

Parametri utilizzati

- **Range di accettazione**: 0 - 16;
- **Range ottimale**: 0 - 10.

Complessità ciclomatica

Indica il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso del metodo/funzione: i nodi del grafo corrispondono a gruppi indivisibili di istruzioni mentre gli archi connettono due nodi se il secondo gruppo può essere eseguito immediatamente dopo il primo.
È possibile ridurre l'indice di complessità attraverso la suddivisione del metodo/funzione in più parti.

Parametri utilizzati

- **Range di accettazione**: 0 - 10;
- **Range ottimale**: 0 - 6.

È accettato anche un valore più elevato, qualora dovesse influire positivamente sulla velocità di esecuzione.

Livello di annidamento

Indica quante strutture di controllo sono inserite l'una all'interno dell'altra. Un alto livello di annidamento può portare ad una complessità maggiore del codice causando difficoltà nella verifica, comprensione e modifica dello stesso.

Parametri utilizzati

- **Range di accettazione**: 0 - 6;
- **Range ottimale**: 0 - 4.

Grado di accoppiamento

Viene calcolato in base a due indici:

- Accoppiamento afferente: numero di classi esterne al $\text{package}_{\text{G}}$ che dipendono da sue classi interne. Un numero alto indica che troppe classi dipendono da tale package, quindi eventuali modifiche provocherebbero forti ripercussioni sull'esterno. Se il numero è basso il package risulta poco utile;
- Accoppiamento efferente: numero di classi interne al package dipendenti da classi esterne. Un numero alto può essere sintomo di una scarsa progettazione.

Grado di instabilità

Tale metrica viene utilizzata per misurare l'instabilità delle componenti di un sistema basandosi sul grado di accoppiamento sopra descritto. Un valore alto indica alta instabilità e quindi bassa libertà di modifica del codice in quanto ogni modifica ha effetti su più classi. Tale indice viene calcolato con la seguente formula:

$$I = \frac{C_{\text{e}}}{C_{\text{a}} + C_{\text{e}}}$$

dove:

- C_{a}: rappresenta l'accoppiamento afferente;
- C_{e}: rappresenta l'accoppiamento efferente.

Parametri utilizzati

- **Range di accettazione**: 0 - 0.8;
- **Range ottimale**: 0.3 - 0.7.

Chiamate innestate di metodi
Indica quante chiamate innestate di metodi sono inserite l'una all'interno dell'altra. Un alto valore può portare a una saturazione dello stack_{G}.

Parametri utilizzati

- **Range di accettazione**: 0 - 6;
- **Range ottimale**: 0 - 4.

Copertura del codice
Rappresenta la percentuale di codice eseguita durante i test. Maggiore è questo valore, più esaurienti saranno i test e maggiori sono le probabilità di individuare gli eventuali errori.

Parametri utilizzati

- **Range di accettazione**: 80% - 100%;
- **Range ottimale**: 90% - 100%.

Numero di linee per metodo
Indica il numero di $\text{statement}_{\text{G}}$ che compongono un metodo. Se un metodo risulta troppo lungo il suo funzionamento risulterà più complicato da comprendere, quindi può essere opportuno dividerlo in più sotto-funzioni. Un metodo troppo lungo potrebbe addirittura essere sintomo di cattiva progettazione della classe.

Parametri utilizzati

- **Range di accettazione**: ≤ 60;
- **Range ottimale**: ≤ 40.

Validazione W3C
L'applicativo web deve superare il test di validazione offerto da W3C con 0 errori gravi. Sono accettati avvisi ed inesattezze che non compromettano le funzionalità del sito fino a un massimo di 10 per pagina.

Parametri utilizzati

- **Range di accettazione**: 0 - 10 (per pagina);
- **Range ottimale**: 0 - 0 (per pagina).

3.2.4 Tabella riepilogativa

Metriche	Range di accettazione	Range ottimale
Metriche per i processi		
Schedule Variance	$\geq -(preventivo * 5\%)$	≥ 0
Budget Variance	$\geq -(preventivo * 10\%)$	≥ 0
Metriche per i documenti		
Indice Gulpease	40 - 100	50 - 100
Metriche per il codice		
Linee di commento su linee di codice	≥ 0.25	≥ 0.30
Numero di parametri	0 - 8	0 - 5
Numero di campi dati	0 - 16	0 - 10
Complessità ciclomatica	0 - 10	0 - 6
Livello di annidamento	0 - 6	0 - 4
Grado di instabilità	0 - 0.8	0.3 - 0.7
Chiamate innestate di metodi	0 - 6	0 - 4
Copertura del codice	80% - 100%	90% - 100%
Numero di linee per metodo	≤ 60	≤ 40
Validazione W3C	0 - 10 (per pagina)	0 - 0 (per pagina)

Tabella 2: Riepilogo misure e metriche

4 Gestione amministrativa

4.1 Definizione di un errore

Si verifica un errore quando si presenta una delle seguenti condizioni:

- Errore ortografico prodotto all'interno dei documenti;
- Violazione dei vincoli imposti dalle norme tipografiche nei documenti;
- Violazione dei vincoli imposti dalle norme tipografiche nel codice;
- Violazione degli indici delle misure e delle metriche prefissate;
- Violazione dei vincoli di prodotto.

4.2 Comunicazione degli errori

La comunicazione degli errori viene svolta dai verificatori attraverso:

- Slack$_G$ nel caso di errori nella documentazione;
- La creazione di "Issue" su GitHub$_G$ nel caso di errori nei file di codice.

4.3 Risoluzione degli errori

Ogni qualvolta vengono riscontrati degli errori, il team si riunisce per decidere come intervenire per risolverli. La risoluzione può quindi essere assegnata ad un solo membro del gruppo o ad un sottoinsieme ristretto di esso.